305-CD-011-001

EOSDIS Core System Project

Release A SDPS Data Processing Subsystem Design Specification for the ECS Project

July 1995

Hughes Information Technology Corporation
Landover, MD

Release A SDPS Data Processing Subsystem Design Specification for the ECS Project

July 1995

Prepared Under Contract NAS5-60000
CDRL Item 046

SUBMITTED BY

_Parag N. Ambardekar /s/___________________7/28/95__
Parag Ambardekar, Release A CCB Chairman Date
EOSDIS Core System Project

Hughes Information Technology Corporation
Landover, Maryland

This page intentionally left blank.

Preface

This document is one of sixteen comprising the detailed design specifications of the SDPS and CSMS subsystem for Release A of the ECS project. A complete list of the design specification documents is given below. Of particular interest are documents number 305-CD-004, which provides an overview of the subsystems and 305-CD-018, the Data Dictionary, for those reviewing the object models in detail. A Release A SDPS and CSMS CDR Review Guide (510-TP-002) is also available.

The SDPS and CSMS subsystem design specification documents for Release A of the ECS Project include:

305-CD-004	Release A Overview of the SDPS and CSMS Segment System Design Specification
305-CD-005	Release A SDPS Client Subsystem Design Specification
305-CD-006	Release A SDPS Interoperability Subsystem Design Specification
305-CD-007	Release A SDPS Data Management Subsystem Design Specification
305-CD-008	Release A SDPS Data Server Subsystem Design Specification
305-CD-009	Release A SDPS Ingest Subsystem Design Specification
305-CD-010	Release A SDPS Planning Subsystem Design Specification
305-CD-011	Release A SDPS Data Processing Subsystem Design Specification
305-CD-012	Release A CSMS Segment Communications Subsystem DesignSpecification
305-CD-013	Release A CSMS Segment Systems Management Subsystem Design Specification
305-CD-014	Release A GSFC Distributed Active Archive Center Implementation
305-CD-015	Release A LaRC Distributed Active Archive Center Implementation
305-CD-016	Release A MSFC Distributed Active Archive Center Implementation
305-CD-017	Release A EROS Data Center Distributed Active Archive Center Implementation
305-CD-018	Release A Data Dictionary for Subsystem Design Specification
305-CD-019	Release A System Monitoring and Coordination Center Implementation

Object models presented in this document have been exported directly from CASE tools and in some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (ECS) at URL http://edhs1.gsfc.nasa.gov.

This document is a contract deliverable with an approval code 2. As such, it does not require formal Government approval, however, the Government reserves the right to request changes within 45 days of the initial submittal. Once approved, contractor changes to this document are handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by document change notice (DCN) or by complete revision.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Corporation
1616 McCormick Drive
Landover, MD 20785

Abstract

This document describes the Release A Detailed Design for the SDPS Data Processing Subsystem. It defines the Data Processing Subsystem computer software and hardware architectural design, as well as the subsystem design based on Level 4 requirements. The subsystem is divided into 3 CSCIs and 3 HWCIs:

a. Processing (PRONG) CSCI

b. Science Data Processing Toolkit (SDPTK) CSCI

c. Algorithm Integration and Test (AITTL) CSCI

d. Science Processing (SPRHW) HWCI

e. Algorithm Quality Assurance (AQAHW) HWCI

f. Algorithm Integration and Test (AITHW) HWCI

Keywords: Processing, SDPS, PDPS, Release A, OMT, Scheduling, AI&T, AutoSys, AutoXpert

This page intentionally left blank.

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Final
iii through xx	Final
1-1 and 1-2	Final
2-1 and 2-2	Final
3-1 through 3-8	Final
4-1 through 4-212	Final
5-1 and 5-2	Final
6-1 and 6-2	Final
7-1 through 7-58	Final
8-1 through 8-20	Final
9-1 and 9-2	Final
10-1 and 10-2	Final
A-1 through A-28	Final
AB-1 and AB-2	Final

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
305-CD-011-001	Final	July 1995	95-0474

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

2. Related Documentation

3. DPS - Data Processing Subsystem

4. PRONG - Processing CSCI

5. SDPTK - Science Data Processing Toolkit CSCI

6. DPREP - Science Data Pre-Processing CSCI

7. AITTL - Algorithm I&T CSCI

8. SPRHW - Science Processing HWCI

9. QAHW - Algorithm Quality Assurance HWCI

10. AITHW - Algorithm Integration & Test HWCI

Figures

Tables

This page intentionally left blank.

1. Introduction

1.1 Identification

This Release A SDPS Data Processing Subsystem Design Specification for the ECS Project, Contract Data Requirement List (CDRL) Item 046, with requirements specified in Data Item Description (DID) 305/DV2, is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000. This publication is part of a series of documents comprising the Science and Communications Development Office design specification for the Communications and System Management segment (CSMS) and the Science Data Processing Segment (SDPS) for Release A.

1.2 Purpose and Scope

The Release A SDPS Data Processing Subsystem Design Specification defines the progress of the design. It defines the Data Processing Subsystem computer software and hardware architectural design, as well as subsystem design based on Level 4 requirements.

This subsystem is on an incremental development track. It is released and reviewed in the form of Evaluation Packages (EP), and is therefore not part of the formal Release A Critical Design Review. The overview material for these components has been included in this document for information purposes only.

This document reflects the June 21, 1995 Technical Baseline maintained by the contractor configuration control board in accordance with the ECS Technical Direction No. 11 dated December 6, 1994.

1.3 Status and Schedule

This submittal of DID 305/DV10 meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. A previous version of this submittal was reviewed during the CSMS Preliminary Design Review (PDR) and reflects changes to the design which resulted from that review. The PDR also triggered a number of follow up actions in response to Review Item Discrepancies (RID) the results of which have been incorporated into this Critical Design Review (CDR) version of this document.

1.4 Organization

The document is organized to describe the Release A SDPS Data Processing subsystem design as follows:

Section 1 provides information regarding the identification, scope, status, and organization of this document.

Section 2 provides a listing of the related documents, which were used as source information for this document.

Section 3 provides an overview of the subsystem, focusing on the high-level design concept. This provides general background information to put the Data Processing subsystem into context.

Section 4 contains the design and structure of the Processing (PRONG) computer software configuration item (CSCI). One of the CSCIs comprising the Data Processing subsystem.

Section 5 contains the structure of the Science Data Processing Toolkit (SDPTK) computer software configuration item (CSCI). This section contains references to other design documents where the SDP Toolkit design has been represented in more detail. One of the CSCIs comprising the Data Processing subsystem.

Sections 6 contains the structure of the Science Data Pre-Processing (DPREP) computer software configuration item (CSCI) comprising the Data Processing subsystem. This CSCI has been consumed into the PRONG CSCI.

Sections 7 contains the structure of the Algorithm Integration & Test (AITTL) computer software configuration item (CSCI). One of the CSCIs which comprise the Data Processing subsystem.

Section 8 contains the structure of the Science Processing (SPRHW) hardware configuration item (HWCI). One of the HWCIs of the Data Processing subsystem.

Section 9 contains the structure of the Algorithm Quality Assurance (AQAHW) hardware configuration item (HWCI). One of the HWCIs of the Data Processing subsystem.

Section 10 contains the structure of the Algorithm Integration and Test (AITHW) hardware configuration item (HWCI). One of the HWCIs of the Data Processing subsystem.

The section Abbreviations and Acronyms contains an alphabetized list of the definitions for abbreviations and acronyms used in this document.

2. Related Documentation

2.1 Parent Documents

The parent document is the document from which the scope and content of this Data Processing Subsystem Design Specification is derived.

194-207-SE1-001	System Design Specification for the ECS Project

2.2 Applicable Documents

The following documents are referenced within this Data Processing Subsystem Design Specification, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document.

209-CD-001-001	Interface Control Document Between EOSDIS Core System (ECS) and the NASA Science Internet
209-CD-002-001	Interface Control Document Between EOSDIS Core System (ECS) and ASTER Ground Data System
209-CD-003-001	Interface Control Document Between EOSDIS Core System (ECS) and EOS-AM Project for AM-1 Spacecraft Analysis Software
209-CD-004-001	Data Format Control Document for the Earth Observing System (EOS) AM-1 Project Data Base
209-CD-005-002	Interface Control Document Between EOSDIS Core System (ECS) and Science Computing Facilities (SCF)
209-CD-006-002	Interface Control Document Between EOSDIS Core System (ECS) and National Oceanic and Atmospheric Administration (NOAA) Affiliated Data Center (ADC)
209-CD-007-002	Interface Control Document Between EOSDIS Core System (ECS) and TRMM Science Data and Information System (TSDIS)
209-CD-008-002	Interface Control Document Between EOSDIS Core System (ECS) and the Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC)
209-CD-009-002	Interface Control Document Between EOSDIS Core System (ECS) and the Marshall Space Flight Center (MSFC) Distributed Active Archive Center (DAAC)
209-CD-010-001	Interface Control Document Between EOSDIS Core System (ECS) and the Langley Research Center (LaRC) Distributed Active Archive Center (DAAC)
209-CD-011-002	Interface Control Document Between EOSDIS Core System (ECS) and the Version 0 System

305-CD-003-002	Communications and System Management Segment (CSMS) Design Specification for the ECS Project
308-CD-001-004	Software Development Plan for the ECS Project
313-CD-002-002	EOSDIS Core System (ECS) Internal Interface Control Document for Science Data Processing Segment (SDPS)
313-CD-003-002	Communications and System Management Segment (CSMS) Internal Interface Control Document for the ECS Project
423-41-03	Goddard Space Flight Center, EOSDIS Core System (ECS) Contract Data Requirements Document
440-TP-008-001 194-00569TPW 194-430TPW-001	The ECS Science and Technology Lab (STL) Prototyping

2.3 Information Documents Referenced

The following documents are referenced herein and amplify or clarify the information presented in this document. These documents are not binding on the content of the Data Processing Subsystem.

Document Number	AutoSys User Manual - v3.1
Document Number	AutoSys Xpert User Guide - Beta

3. DPS - Data Processing Subsystem

3.1 Subsystem Overview

3.1.1 Introduction and Context

The Data Processing Subsystem is the collection of hardware and software components which are responsible for the management of the data processing resources at a provider site. These management responsibilities can be divided into the following general functional areas:

a. Managing the generation of Data Products and the operational environment used to produce these products.

b. Providing an Algorithm Integration and Test Environment for the introduction of science software into the EOSDIS environment.

The Data Processing Subsystem supports these functional areas through the following mechanisms:

a. It provides a batch processing environment to support the generation of data products. It manages, queues and executes Data Processing Requests on the processing resources at a provider site. A Data Processing Request can be defined as 1 processing job. Each Data Processing Request encapsulates all of the information needed to execute this processing job. Data Processing Requests are submitted from the Planning Subsystem; which in turn have been triggered by arrival of data or or internally through Planning itself (e.g., reprocessing). Data Processing Requests use Product Generation Executives (PGEs) to perform this processing. PGEs will result from the integration and test of delivered science algorithms [ref.: ECS White Paper 193-00118] and also user specific methods into the subsystem. They will be encapsulated in the ECS environment through the SDP Toolkit. The Data Processing subsystem also provides the Operational interfaces needed to monitor the execution of the science software (PGEs). More information on the execution of PGEs, and the support environment being provided to monitor the generation of data products is provided in Section 4, the Processing CSCI.

b. It supports the execution of science algorithms through the SDP Toolkit. The SDP Toolkit is a set of tools developed to standardize and provide a common interface for each science algorithm to the EOSDIS environment.

   The following documents provide guidance on the roles and responsibilities of the SDP Toolkit to support the execution of science software:

333-CD-001-002	SDP Toolkit Users Guide for the ECS Project
193-801-SD4-001	PGS Toolkit Requirements Specification for the ECS Project, FINAL, 10/93 [AKA GSFC 423-06-02]

c. It supports the preliminary processing of data sets, i.e., L0 and ancillary data products, which are required by the science algorithms, but are not in the proper format for use.

d. It provides the Algorithm Integration and Test environment used to integrate new science algorithms, new versions of existing science algorithms and user methods into the EOSDIS environment. The algorithm or method will be acquired by the system through an ingest client which will reflect local site policies on the acceptance of software for integration. Once acquired, the algorithm/method and its associated data files (test, calibration, etc.) will be registered in the local site Configuration Management (CM) system as part of the archival by the Data Server Subsystem. These activities are defined in more detail in Section 7, the Algorithm I&T CSCI.

3.1.2 Data Processing Subsystem Context

A context diagram illustrating the relationships between the Data Processing Subsystem and the other SDPS subsystems is shown in Figure 3.1-1. The key interfaces shown are:

a. The *Planning* interface is responsible for determining what processing activities are required to generate the data products as specified in a Production Request (Standard). These processing activities and associated information are defined and delivered as Data Processing Request(s) to the Data Processing Subsystem. One Production Request may result in 1 or more Data Processing Requests being sent to the Data Processing Subsystem. After the receipt of a Data Processing Request, the Data Processing subsystem will deliver processing status to Planning, when requested. Also, provided by the Data Processing Subsystem in an unrelated off-line activity is information used to plan the execution of a PGE. This information is determined by the Algorithm Integration & Test services and provided to Planning through the adding of data to the PDPS Database.

b. The *Data Server* interface is for requesting access to data required as an input to a PGE and for requesting that generated output data be transferred to the Data Server. Also, The *Data Server* is used to archive PGEs and associated data which require staging.

c. The *Ingest* interface is for requesting access to Level 0 data required as an input to a PGE.

d. The *MSS/LSM* interface provides access to common ECS system management services.

In Table 3.1-1, where an exact number is unavailable, the data volume is estimated as low (less than 1 MB), medium (between 1 MB and 1 GB), or high (greater than 1 GB) per use defined in the frequency column.)

Processing requirements are driven by the frequency of Production Requests and their associated Data Processing Requests which vary by site. Therefore, processing volume and resulting capacity requirements are discussed in the DAAC specific documents of this DID.

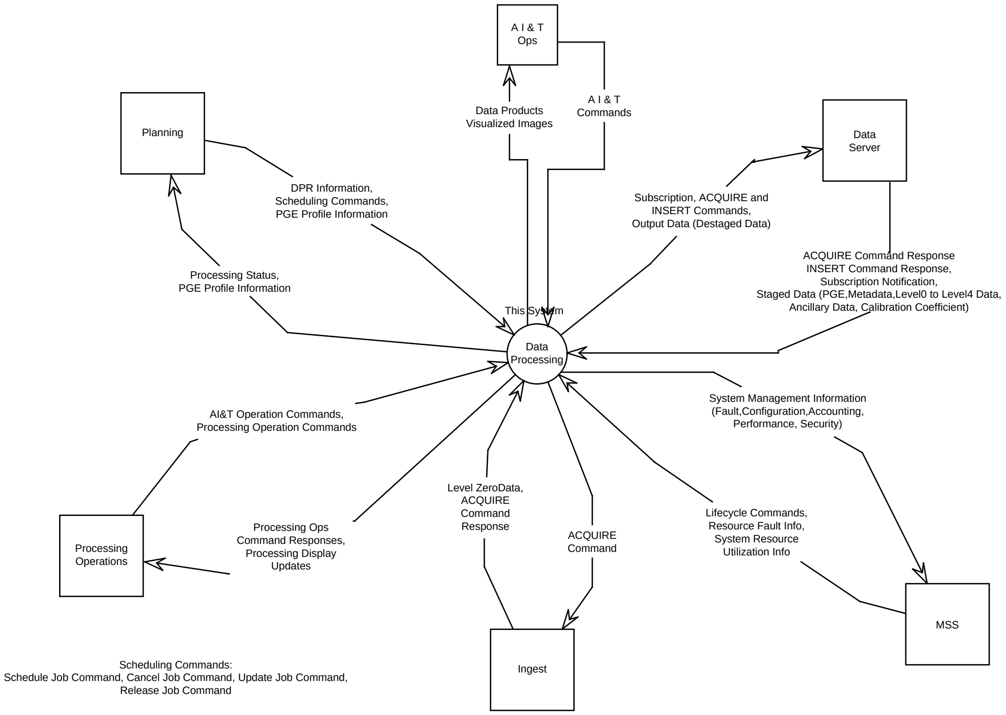


Figure 3.1-1. Data Processing Subsystem Context Diagram

Table 3.1-1. Subsystem Interfaces (1 of 3)

Source	Destination	Data Types	Data Volume	Frequency
Data Processing	Data Server	Acquire Commands (Staging Requests)	low	as required for processing
Data Processing	Data Server	Subscription Requests	low	as required for processing
Data Processing	Data Server	standard products	high	as required for processing
Data Processing	Data Server	Metadata	high	as required for processing
Data Processing	Data Server	q/a data	medium	as required for processing
Data Processing	Data Server	Science Algorithms	low	as requested
Data Processing	Planning	Schedule Job Command Response	low	In Response to Schedule Job Command
Data Processing	Planning	Cancel Job Command Response	low	In Response to Cancel Job Command

Table 3.1-1. Subsystem Interfaces (2 of 3)

Source	Destination	Data Types	Data Volume	Frequency
Data Processing	Planning	Release Job Command Response	low	In Response to Release Job Command Response
Data Processing	Planning	Update Job Command Response	low	In Response to Release Job Command Response
Data Processing	Planning	Processing Status	low	In Response to Request For Job Status
Data Processing	Planning	PGE Profile Information	low	as required
Data Processing	MSS	System Management Information (Accounting, Scheduling, Fault, Accountability, Security, and Performance)	low	as required
Data Processing	Operations	Processing Operations Information	low	in response to request
Data Processing	Operations	AI&T Operations Information	low	in response to request
Planning	Data Processing	Schedule Job Command	low	in response to activating a Plan
Planning	Data Processing	Cancel Job Command	low	in response to activating a Plan
Planning	Data Processing	Release Job Command	low	in response to activating a Plan
Planning	Data Processing	Update Job Command	low	in response to activating a Plan
Planning	Data Processing	Processing Status	low	in response to activating a Plan.
Planning	Data Processing	PGE Profile Information	low	as required
Data Server	Data Processing	Standard Products	high	in response to staging request
Data Server	Data Processing	Metadata	medium	in response to staging request
Data Server	Data Processing	Ancillary Data	high	in response to staging request
Data Server	Data Processing	Calibration Data	medium	in response to staging request
Data Server	Data Processing	Orbit/Attitude Data	medium	in response to staging request
Data Server	Data Processing	Algorithms	low	in response to staging request

Table 3.1-1. Subsystem Interfaces (3 of 3)

Source	Destination	Data Types	Data Volume	Frequency
Data Server	Data Processing	Science Software Delivery	low	as required
MSS	Data Processing	resource fault information	low	as required
MSS	Data Processing	resource performance utilization information	low	as required
Operations	Data Processing	Processing Operations Commands	low	as required
Operations	Data Processing	AI&T Operations Commands	low	as required

3.2 Subsystem Overview

The following sections summarize the hardware and software structure of the Data Processing Subsystem and provide the design rationale used during the Detailed Design process.

3.2.1 Subsystem Structure

The Data Processing Subsystem is divided into the following CSCIs and HWCIs:

- PRONG – Processing CSCI—The Processing CSCI provides the services required to manage and monitor the Science Data Processing environment which is used to generate data products using Science Software (PGEs) provided by the instrument teams. These services are discussed further in Section 4, the Processing CSCI.
- SDPTK – SDP Toolkit CSCI—The SDP Toolkit CSCI provides a set of software libraries which are used to integrate Science Software into the EOSDIS environment. By promoting the POSIX standard, these libraries allow the Science Data Processing environment to support the generation of data products in a heterogeneous computer hardware environment. The following documents provide guidance on the roles and responsibilities of the SDP Toolkit to support the execution of science software:

  333-CD-001-002 SDP Toolkit Users Guide for the ECS Project, 11/94

  193-801-SD4-001 PGS Toolkit Requirements Specification for the ECS Project, FINAL, 10/93 [AKA GSFC 423-16-02]

  - AITTL – Algorithm I&T CSCI—The Algorithm I&T CSCI is a set of tools which are used to integrate and test new science software, new versions of science software and user methods into the Science Data Processing operational environment. These services are discussed further in Section 7, the Algorithm I&T CSCI.
- SPRHW – Science Processing HWCI—The Science Processing HWCI is the collection of hardware resources being provided to support the generation of data products in a secure, monitored environment. These services are discussed further in Section 8, the Science Processing HWCI.

- AQAHW – Algorithm QA HWCI—The Algorithm QA HWCI is the collection of hardware resources being provided to support DAAC manual quality assurance activities. These services are discussed further in Section 9, the Algorithm QA HWCI.
- AITHW – Algorithm Integration & Test HWCI—The Algorithm Integration & Test HWCI is the collection of hardware resources being provided to support the integration and testing of Science Software in a secure, monitored environment. These services are discussed further in Section 10, the Algorithm Integration & Test HWCI.

3.2.2 Subsystem Design Rationale

The division of the subsystem into its current CSCIs and HWCIs has been driven by many different factors, but two are of particular importance:

a. Science algorithm development support.

b. Efficient use of the Science Data Processing Environment.

Because of the need to support the development of science algorithms, the SDP Toolkit is an early deliverable. Therefore, a need arose to group this software at a CSCI level for accountability reasons. The SDP Toolkit is an extensive software library with multiple language bindings. It is being incorporated into the science software to avoid the duplicate development of commonly used tools, in addition to providing a standard interface for integration of science software into the ECS. In particular, the library provides interfaces for accessing ECS data products, product metadata, ancillary data and processing parameter information. This Toolkit was needed by the Instrument Teams at an early date to support development, testing and early integration of science software.

The Processing CSCI was developed to provide a means of monitoring and managing the batch generation of scientific data products. DAAC Science Data Processing is not a real-time, automatically invoked, single data stream, uniform process such as spacecraft telemetry processing. Science data must be collected, stored, prepared, and "batched" for generation of lower-level data products. Processing occurs post (Satellite) pass and must be planned on the basis of data quality and availability, and the availability of appropriate system resources. The appropriate planning of the generation of data products is performed by the Planning Subsystem (see Planning Subsystem Detailed Design Specification for more details.). In support of Science Data Processing, the Processing CSCI provides the following capabilities to SDPS:

1. Provides effective resource management for efficient use of hardware resources.
2. Supports error recovery mechanisms to allow the generation of data products in an operator attended or unattended processing environment.
3. Provides mechanisms to support many different Science Processing hardware configurations which are needed to support the differing aspects of data processing which occurs at each DAAC site.
4. Provides data pre-processing support functions for the introduction of Level 0, Ancillary data, Orbit and Attitude data into the ECS. This data may be received in formats which are not usable by the science software, and therefore, must undergo reformatting and/or other types of preparation.

The Algorithm I&T CSCI, on the other hand, is the grouping of tools and support functions related to the integration of science software and user methods into the ECS. This CSCI is a standalone function and requires special attention to support a secure, standalone Algorithm I&T environment.

The division of the Hardware CSCIs was driven by the need to support a Science Data Processing environment which would not be affected by the Algorithm I&T environment. The hardware to support the Science Data Processing Environment is grouped as the Science Processing HWCI. The Algorithm I&T environment is grouped as the Algorithm Integration & Test HWCI. The Algorithm QA HWCI was created to support DAAC manual quality assurance activities. This HWCI must support the display of generated data products and the updating of quality assurance metadata associated with a data product.

3.2.2.1 Performance

The Data Processing subsystem design has been prepared to meet the overall performance objectives of the TRMM Release period. The performance for the subsystem for a given workload is determined by both the hardware and the software designs for the system. The workload for the TRMM Release period as estimated from the Technical Baseline is relatively modest. The hardware platforms to be provided at the TRMM Release period will, therefore, be sized to meet the workload that the system will be exposed to during the EOS-AM1 Release period, which is significantly more challenging.

The performance of the PRONG CSCI of the Processing subsystem will be strongly affected by the COTS products, Platinum Technology's AutoSys and AutoXpert scheduling tools, that have recently been procured. Prior to the release of the request for proposals for this tool, several such packages, including AutoSys were evaluated by HAIS. Certain performance measures were taken, and this information has been used in the design of the platforms to be used to support this element of the Processing subsystem. Informal prototyping activities of the AutoSys and AutoXpert packages have been initiated and will continue after CDR, both to evaluate details of the COTS implementation and to evaluate the performance of the COTS package. Results of this evaluation will be analyzed and may influence the details of the hardware sizing for the platforms that will support the PRONG CSCI.

Details concerning the performance of the platforms supporting the PRONG CSCI, including considerations of the RMA requirements, are included in the sections in this document devoted to the associated hardware platforms. It should be noted that the design for the PRONG CSCI and associated software reflects the high reliability option for AutoSys and AutoXpert, including redundant processors and high reliability Sybase servers. This will insure that the mean time to restore service is within the 30 minute recovery time required.

In addition, the design of the Planning and Data Processing subsystem software have been prepared to minimize needless disk I/O. This is accomplished by retaining on local disk the output files of one PGE, and scheduling the execution of another PGE that use those files to occur soon afterwards. In this way, needless disk I/O and network loading is avoided, insuring that the ECS performance goals can be met with the least cost.

This page intentionally left blank.

4. PRONG - Processing CSCI

4.1 CSCI Overview

The Processing CSCI is responsible for the initiation, managing, and monitoring of the execution of science software algorithms. These science software algorithms are identified to the Processing CSCI through a PGE. From the ECS Glossary, a PGE is defined as "a set of one or more compiled binary executables and/or command language scripts; it is the smallest unit that can be scheduled for the Product Generation System (PGS, now the Planning and Data Processing Subsystems) processing." A PGE is equivalent to one processing job which requires the use of the Data Processing Subsystem's hardware and software resources. Generally, a PGE will be used for the generation of ECS Data Products, but a PGE may be defined to perform other types of processing, such as preprocessing of input data or the quality assurance processing of generated Data Products. PGEs that generate data products and perform quality assurance are provided by the Instrument Teams and algorithm developers. The Processing CSCI is informed of the required execution of a PGE through a Data Processing Request message received from the Planning CSCI. The Processing CSCI will not initiate the execution of a PGE until all necessary data required as an input to the PGE is available. Available in this context means that the data exists at a Data Server, not necessarily the on-site Data Server.

In support of the execution of the science software algorithms, the Processing CSCI has the following responsibilities:

a. Manages the science software algorithm execution process.

b. Manages Science Data Processing computer hardware resources efficiently.

c. Manages the flow of data required to execute a science software algorithm.

d. Manages the flow of data produced by the execution of a science software algorithm.

e. Provides an Operational interface to allow monitoring of processing status, and manual intervention, when necessary, into Science Data Processing operations environment, including processing queue control.

f. Provides an Operational interface to support the quality assurance of generated data products.

4.1.1 Processing CSCI Design Rationale

The Processing CSCI detailed design, as presented, is meant to provide a framework to build a robust science data processing system. This detailed design has taken the concepts expressed in the Data Processing Subsystem sections of the SDPS System Design Specification and the SDPS Preliminary Design Specification and expanded, improved, and modified these concepts to reflect the current state of the Processing CSCI design.

This detailed design represents a significant step in providing a basis to continue into the Implementation phase. By developing the use-case scenarios (see Section 4.5, Processing CSCI Dynamic Model), the design has established the boundaries of the Processing CSCI software, allowed the identification of problem areas, and the resolution of these problem areas. These scenarios will now be used as guidance for implementation.

The Processing CSCI design rationale has been influenced by a set of design drivers. These design drivers have greatly influenced the decisions which have been made as the design has matured since the System Design Specification and the Preliminary Design Specification for the ECS Project. The following design drivers have been identified:

a. Separation of Planning and Processing—The roles and responsibilities of the Planning CSCI and Processing CSCI have been separated to support future evolving aspects of these functions.

b. Data Driven System—The Processing CSCI does not activate a Data Processing Request which requires execution until data is available to support the execution of the PGE associated with Data Processing Request.

c. Priority Driven Data Processing—As a Data Processing Request is input into the Processing CSCI, this Data Processing Request has an assigned priority which is determined in the Planning CSCI. This priority is based on the production rules used to generate the production plan which is currently active. This priority is reflected by the production policies at a DAAC.

d. Development of user interfaces—The amount of support to provide the different classes of users, i.e., Operations, Instrument Teams, and remote users, has influenced how information is to be provided to each type of user. The need to support a secure operations environment is quite evident.

e. Exception handling—The capabilities to support recovery from the faults associated with the failure of a resource or a PGE are provided to support efficiency in the science data processing environment.

g. Extent of automation vs. manual supported operations—To support an efficient Science Data Processing environment, a number of decisions have been made in the design to reflect the need to increase the amount of automated decision making required. This is driven by the need to support the generation of data products in an attended (by Operations staff) or unattended mode.

h. Dynamic aspects of Planning and Processing—Both Planning and Processing software must have the capability to react to real-time events, i.e., PGE failures, resource failures, etc., which have affected the active production plan and the queue of Data Processing Request Jobs awaiting execution in Processing.

Each of these design drivers has impacted the framework of the Processing CSCI design. To emphasize their impact in the different functional areas affecting the design, a description of the design in certain areas and the underlying factors affecting the design has been provided. These descriptions are in the following sections:

a. The division of the Planning and Processing CSCIs

b. Resource Management

c. Quality Assurance

d. Processing Error Architecture

e. Processing/Planning Interface

f. Processing/MSS Interface

g. Processing/Data Server Interface

The information provided in the following sections on the above listed topics is meant to summarize important aspects of the Processing CSCI detailed design. More information on these areas and the resulting impact to the detailed design is contained in the Section 4.5, Processing CSCI Dynamic Model, which contains a set of scenarios used to define the roles and responsibilities of the Processing CSCI.

4.1.1.1 The Division of the Planning and Processing CSCIs

The division of the Planning and Processing CSCIs was influenced by the following factors:

a. Follows Client/Server Architecture—Planning acts as the Client, and Processing acts as the Server. Planning is responsible for developing a Production plan and informing Processing which activities must be executed as listed in the Production plan.

b. Increases Fault Tolerance of the Planning and Data Processing Subsystems—By separating the Planning services from the Processing services, an increase in the fault tolerance of the Planning and Data Processing Subsystem software is achieved. The failure of the Planning subsystem will not cause an immediate breakdown in production processing. Processing will be able to continue with production processing until the processing queues are depleted. Also, a failure of the Processing CSCI or Processing Hardware will not cause the Planning subsystem to fail. Although the flow of information from Planning to Processing and Processing to Planning will be interrupted, this should not affect the overall production plan.

c. Evolvability of the roles and responsibilities of Planning as ECS becomes operational—The separation of Planning services from Processing services allows for different configurations of Planning services and Processing services. For example, there may be some special event which would result in the Planning services of a DAAC creating a Production plan for itself as well as other DAACs. By separating Planning from Processing, this becomes an easier problem to resolve.

Each of the above items has influenced the separation of Planning services from Processing services. This separation will support the general concept of the changing roles of Planning and Processing as ECS matures.

4.1.1.2 Resource Management

The Resource Management capabilities as presented in the Processing CSCI detailed design have been influenced by the following design drivers:

a. Efficient use of computer resources is required to support Production Processing—The Processing CSCI is responsible for the allocation of resources, i.e., disk space, memory, and CPU, to execute a PGE. To perform this role, Processing relies on MSS to provide up to date (pseudo-real-time) information on the health and overall availability of Data Processing Hardware resources. This information is provided by MSS SNMP monitoring agents which will be located on each science processing resource platform.

The Processing CSCI allocates disk space, memory, and CPU to support the execution of a PGE. The PGE resource profile information for allocating disk space, memory and CPU are established during the AI&T time frame and will undergo updates as the science software matures in the science data processing environment

Before a PGE begins execution, all required resources must be available. This means that enough disk space and sufficient CPU must exist to support execution. These resources will be allocated to only support the execution of a given PGE. Also, as required through the Planning CSCI, all input data is available before initiating the PGE. This will alleviate any deadlock situation where a PGE is awaiting some input data file or awaiting the use of a given resource currently in use by another PGE. Also, as determined through ECS dynamic modeling results, it seems that a majority of PGEs, as currently defined, will be executed individually on a CPU. This has been shown to provide an optimal solution. Even though this is the current direction, no Processing CSCI or COTS decision precludes the availability of the time-sharing of CPUs for multiple PGEs.

Also, during execution monitoring, Processing will monitor the use of the allocated disk space, memory, and CPU as a method of fault detection.

b. Automated recovery from resource faults—Processing plays a limited role in resource fault recovery. Processing is responsible for protecting the state data maintained in the Processing software and for insuring that the execution of a PGE can resume upon the recovery of a resource. When a resource has failed, the Processing CSCI will inform MSS of the failure and update the resource management information associated with the failed resource to indicate that the resource can not be allocated for processing. Also, MSS will inform Processing that a resource is unavailable and when the resource is available again for processing. This interface is intended to be two way.

c. Reporting of resource fault and other information—Processing supports fault reporting in two distinct ways:

1. During PGE execution—While a PGE is executing, information is collected from the execution of the science software through the use of a Status Message File. Processing will inform MSS about resource-specific faults. To recover from a resource fault, the Processing CSCI may determine if the PGE can be executed on another resource which is capable of supporting the execution of the PGE. This will be done by allowing the Data Processing Request Jobs to re-queue into the queue supporting the other resource. For a science software fault, Processing will use return code information provided by the PGE to determine the necessary steps to take for recovery purposes.

2. Processing SW specific—If an event for which Processing is responsible (i.e., data staging, data destaging, execution environment monitoring) indicates a possible resource-specific fault, Processing will inform MSS.

d. Logging of resource fault and other information—Resource fault and usage information will be logged for accounting, performance, and accountability purposes. The resource usage characteristics for a PGE will be updated and used for future Production planning purposes. Fault information will be used to determine if faults are re-occurring frequently.

The logging of resource management data is performed by MSS. Processing will assist by providing system management information to MSS.

4.1.1.3 Quality Assurance

This section briefly describes quality assurance and the support provided by SDPS. The support provided for DAAC Manual quality assurance activities by the Processing CSCI is still at a conceptual level and is evolving as requirements generated through Instrument Team input are understood. Four types of quality assurance will be supported by SDPS.

a. In-Line Q/A—This service is provided through the production environment. This is the automated quality assurance processing which can be provided by a PGE executing a quality assurance algorithm. At this time, a PGE could exist which only performs automated quality assurance on a generated product, or the quality assurance algorithm may be one part of a PGE which generates a data product. These quality assurance PGEs will be provided by Instrument Team algorithm developers.

b. DAAC Manual Q/A—DAAC manual Q/A capabilities can be supported through the use of the Data Server subscription service. The DAAC Q/A position would have to manually subscribe to the desired data product. The Data Server would provide notification to the DAAC Q/A monitor when a selected data product has been generated and is available for review. Also, this could allow DAAC Q/A to be de coupled from the generation and initial storage of the data product. DAAC Q/A activities would not necessarily have to occur when the product is generated.

c. SCF Q/A—This service is provided through use of the Data Server subscription service. A user who wishes to perform Q/A on a product can request a copy of the product whenever the product is generated by placing a subscription for the data product.

d. User Q/A—User quality assurance activities occur as a data product is used to support research. The ability of the remote user to update quality assurance information is necessary. For existing data, a user can query for and request data directly.

By using the Data Server subscription service, the Processing CSCI lessens the dependency on man-in-the-loop activities affecting the generation of data products. Processing SW is being designed under the assumption that Data Processing must be able to generate products while in an attended or unattended mode. If the Q/A position is manned when the product is generated, the product can be reviewed while still on the Data Processing subsystem storage device. Otherwise, the Q/A position may have to request the product from the Data Server. To allow DAAC manual Q/A to work efficiently, Data Processing will require knowledge of what type of Q/A the product will require. As the requirements for DAAC manual quality assurance activities become understood and stable, further attempts to automate these activities will be supported.

The Processing CSCI will support an interface for visual display of a data product through the use of data visualization tools, such as EOSVIEW and IDL. Processing will also support an interface to allow for the update of the Q/A metadata which is associated with a data product.

The design approach was chosen for the following reasons:

a. Based on the Data Processing design approach of allowing for manned and unmanned modes of operations.

b. Follows Q/A approach for making products available for SCF review. No new paradigm to support DAAC manual Q/A has been introduced. This leads to maximum reuse of already developed software capabilities.

c. Allows man-in-the-loop Q/A activities at the DAAC without serious impact on the processing resources. This approach would not require manual intervention before processing of other data products continues. If Q/A position is unmanned, the data products requiring Q/A review are stored at the Data Server and await review.

4.1.1.4 Processing Error Architecture

This section provides a brief description of the Error Architecture approach adopted by the Processing CSCI and follows the common error handling approach adopted across all SDPS applications. Error Architecture refers to the mechanisms used for error detection, reporting and recovery that are incorporated into the design of the Processing CSCI. It provides details on how the Processing CSCI will react when an error or exception, i.e., hardware or software, occurs during steady-state processing of a Data Processing Request and the execution of a PGE.

This Processing CSCI Error Architecture approach was influenced by the following factors:

a. Detection of different types of errors, i.e., Science Software (PGE) execution errors, Processing Hardware errors, and Processing Software errors.

1. The detection of science software errors which occur during the execution of a PGE are captured by the SDP Toolkit, and are propagated to the Processing CSCI through the use of a PGE return code status. The Processing CSCI will have knowledge of the error associated with a given return code, and as a result of this return code, may initiate a corrective measure, such as alerting the operations staff or restarting the PGE job with updated diagnostic flags to initiate the capture of detailed diagnostic information. The definition of these return codes and the resulting corrective measures are an ongoing activity. Dialogue with Release A instrument teams, i.e., CERES, have led to some initial definitions. These are currently being defined with more detail and will be supplied to all instrument teams for more feedback.

   As a result of an unsuccessful return code, the running of dependent Data Processing Request jobs which were dependent on PGE that failed would not be executed. This information would be made available to the Operations staff through the use of the Processing Operations Interface. Please see Section 4.1.3, COTS Selection, for more information on the Processing Operations Interface. The Processing Operations Interface is provided by the selected COTS product.

2. Although MSS will also detect science processing hardware errors, the detection of Processing HW and Processing SW errors can occur in the Processing SW. Processing HW error detection will be supported by a group of MSS interfaces used to support fault tolerance and provide resource management information.

b. These errors must be reported to different user classes depending on the nature of the error, i.e., IT/Algorithm Developer, DAAC Operations personnel. The IT/Algorithm Developer users are interested in the detailed error information associated with the execution of a PGE for the purposes of debugging a problem. This information will be logged in Status Message Files which are created and maintained during PGE execution by the SDP Toolkit.

While the DAAC Operations personnel is interested in whether a PGE was successfully executed, the detailed error information will not be provided for review, unless requested. Rather, the DAAC Operations personnel will be alerted to the unsuccessful processing, and the alert information will be captured in the processing log for later use.

c. Recovery actions from errors will differ on the type of error. The recovery from an error will depend on the type of error. In almost all severe error cases, the recovery action will be to terminate the event which has suffered the error. This approach is also required to support minimal human interaction by the Operations staff. This helps support the concept of Production Processing occurring in an unattended mode.

d. Human interaction requirements to support Production Processing must be minimized. To support this, almost all errors will be logged and the Operations staff will be alerted. Depending on whether Production Processing is occurring in an attended operations mode or unattended operations mode, the operations staff will have the capability to manually intervene to correct the error condition. Otherwise, the activity will be terminated by Processing, and a new processing activity will be initiated.

e. Isolation of errors so as not to affect other Processing activities. Science SW processing is terminated to isolate the cause of the error. Also, Processing HW may be taken off-line to correct the resource problems to support the isolation of the error condition.

4.1.1.5 Processing/Planning Interfaces

This section provides a brief description of the relationship between the Planning and Processing CSCIs.

The interface between the Planning and Processing CSCIs occurs through a common shared database, known as the PDPS Database. The PDPS Database is an RDBMS, SYBASE, and contains all data which is persistent to applications associated with Planning, Processing, and Algorithm Integration and Test. The decision to share a common database was driven by the many common data structures which were apparent in the Preliminary Design Specifications developed for the Planning and Data Processing Subsystems.

When a Data Processing Request is planned for execution, as represented in the activated production plan which is created and maintained by Planning CSCI components, the knowledge of this Data Processing Request is transferred into the Processing CSCI domain. This involves adding the definition of a series of jobs representing the Data Processing Request to the COTS product being used to manage production by the Processing CSCI. Please see Section 4.1.3 for more information on the selected COTS product. As these jobs run, the Planning CSCI is capable of polling on the COTS product to retrieve status information for a given job.

4.1.1.6 Processing/MSS Interfaces

This section provides a brief description of the relationship between the Processing and MSS CSCIs. A high-level description of the interfaces between MSS and Processing is provided.

The Processing CSCI is dependent on the MSS to provide life cycle services. These services consist of information related to the following activities:

a. Startup of the Processing CSCI software components.

b. Shutdown of the Processing CSCI software components.

MSS provides proxy agents which will be used to communicate startup and shutdown commands to all ECS applications. It is currently envisioned that the MSS proxy agent initiates the PDPS Database. The startup of the PDPS Database will then initiate the startup of the Processing CSCI COTS components. The COTS components will then initiate the custom Processing CSCI components on an as needed basis.

Also, MSS provides mechanisms which enable the Processing CSCI components to provide system management information, such as system-wide event information, resource fault information, and ECS application fault information, to MSS for logging purposes and to initiate system-wide error recovery activities.

In support of system resource configuration management, MSS provides resource configuration information to the Processing CSCI which allows the Processing CSCI to logically manage the allocation of resources to support science data production. MSS will support the monitoring of science software by providing fault isolation and performance tools which provide feedback on the utilization of the science data processing resources.

4.1.1.7 Processing/Data Server Interfaces

The Processing CSCI has an interface to the Science Data Server CSCI to support the staging (Acquiring of data from the Science Data Server CSCI) and destaging (Insertion of data to the Science Data Server CSCI) of data. At this time, this interface is being viewed as a classic client/server interface with the Processing CSCI acting as the client. The Processing CSCI requests the Science Data Server CSCI to stage or destage data as required to support the generation of a data product. The Processing CSCI will inform the Science Data Server CSCI where the data should be staged (what resource) or where the data should be destaged (copied) from. The Science Data Server CSCI uses an FTP PUSH to copy the data to the science processing resources to support staging of data and uses an FTP PULL to copy the data from the science processing resources to support destaging of data. When the Science Data Server CSCI has completed this task, the Science Data Server CSCI informs the Processing CSCI that the data has been staged or destaged successfully. The Processing CSCI is responsible for determining whether data should be deleted from the science processing resources. When data is destaged, it is considered a copy operation, not a move operation.

4.1.2 Processing CSCI Design Modifications since PDR

After the Preliminary Design Specification, a COTS product(s) has been selected which will fulfill the majority of Level 3 and Level 4 requirements. This COTS product has had a tremendous impact on the detailed design as presented here. The following information will summarize these design modifications and their rationale. The selected COTS products are Platinum Technology's **AutoSys** and **AutoXpert.** They will be integrated into PDPS to provide the basis for the monitoring and management of ECS' science data production facility.

All design decisions have been driven by a desire to minimize custom code development, tempered by the need to provide proper encapsulation of the COTS to insure later flexibility of adding or modifying the underlying COTS product as ECS matures and evolves.

As a result of these efforts, some design elements which were mapped to the Planning CSCI at PDR have since moved to the Processing CSCI. This has resulted in a clearer division of the roles and responsibilities of the Planning and Processing CSCIs. As a side effect, the Planning and Processing CSCIs are now more loosely coupled. This will ensure greater flexibility in the future.

The following summarizes the design decisions and provides a top-level view of the current Planning and Data Processing Subsystem Architecture.

1. PDPS will share a common database, i.e., one instance of a SYBASE RDBMS. This will allow PDPS to eliminate the large amount of common persistent data structures which existed in the PDPS preliminary design. For detailed information on the PDPS Database, please refer to Section 4.6.6, PDPS Database CSC, in the Planning Subsystem Preliminary Design Specification
2. The Production Management CSC which was mapped to the Planning CSCI has been divided between the Planning and Processing CSCIs. As presented at PDR, the Production Management CSC provided two important functions; managing of subscription notifications from the Data Server and Ingest and managing the active plan by receiving status feedback from the Processing CSCI. Since the AutoXpert product provides mechanisms for monitoring and managing the active plan, it was decided to encapsulate the COTS products into a single COTS CSC within the PRONG CSCI. This will provide a more consistent and simpler design with fewer interfaces needed between the Planning and Processing CSCI. Therefore, active plan management is now within the Processing CSCI, whereas, the management of subscription notifications remains in the Planning CSCI.
3. The interface between the Planning and Processing CSCIs has been modified. This change involves when a Data Processing Request is made visible to the Processing CSCI. At PDR, the approach amounted to not providing a Data Processing Request to the Processing CSCI until all the data subscriptions, sometimes called data dependencies, were fulfilled for a Data Processing Request. Because of the selection of AutoSys and its capabilities to manage job dependencies, this approach has been changed to consist of all Data Processing Requests being fed into AutoSys at the beginning of the day. The Data Processing Requests which do not have all data dependencies fulfilled would be kept in a "HELD" state until the dependencies are fulfilled. Upon the meeting of all data dependencies, the Planning CSCI would release the job.
4. The software components of the Science Data Pre-Processing CSCI as defined in the Preliminary Design Specification have been mapped into the Processing CSCI or Ingest CSCI, based on what is the optimal location to perform these operations. Within the Processing CSCI, the Science Data Pre-Processing functions have been mapped to a CSC called Data Pre-Processing.

4.1.3 COTS Strategy

This COTS Strategy section summarizes the objectives and technical approach which was taken to determine the optimal COTS solution required to support the job scheduling functions associated with the Processing CSCI management of the science processing resources. There were four objectives which influenced the decision on the type of COTS products selected;

1) Minimize custom code development

To insure an adequate solution for the Release A time frame, the COTS solution must minimize the amount of custom code development which is required to provide the complete Planning and Data Processing Subsystem software solution. This is necessary to mitigate the schedule risk associated with Release A which because of time constraints, will not support a large growth in custom code development.

2) Reach COTS decision to support Release A Detailed Design and Implementation Phase.

   An early decision on the COTS product was required to insure that the required modifications to the Planning and Processing CSCI design could be made to support CDR and to ultimately support the Release A software development schedule.

3) Ease of integration into ECS software applications.

   The selected COTS product must support command line or API style interfaces to support the extensive integration efforts which must occur to meet ECS requirements.

4) Scalability to Release B processing load.

   The selected COTS product must be capable of supporting large numbers of jobs, i.e., 20,000, per day. This is necessary to support the Release B processing load.

The adopted technical approach consisted of the following steps:

1) Collecting information about the different types of COTS which could be used to support ECS Planning and Processing functions;
   a) Vendor teleconferences and meetings
   b) Customer teleconferences
   c) Vendor site visits
2) Determining different types of software architecture given the COTS products and the unique ECS Planning and Processing requirements.
3) Analyzing the different classes of COTS packages to determine viability in ECS given the previously defined objectives.

The information gained in this process was used during the preparation of the RFP (Request For Proposal). This RFP was divided into mandatory, optional, and implementation features sections. The responses to these proposals were analyzed and scored based on the information provided by the vendor for each of these sections.

4.1.4 COTS Selection

The following sections provide information on the COTS products selected to support the Processing CSCI in performing management and monitoring activities associated with ECS' science data production environment. The product selected was Platinum Technology's AutoSys and AutoXpert products. A summary of the AutoSys' capabilities and a scenario which explains the set of actions taken to initiate a job is provided.

4.1.4.1 Platinum Technology's AutoSys

AutoSys is a job scheduling software application used to automate operations in a distributed UNIX environment. AutoSys performs automated job control functions required for scheduling, monitoring, and reporting on jobs that reside on any machine attached to a network. In ECS, the machines for which AutoSys will be responsible are the Science Processing hardware resources.

In AutoSys, a job is defined as any UNIX command or shell script plus a variety of qualifying attributes which include conditions specifying when and where the job should be run.

AutoSys provides a complete system solution to support job scheduling. This includes an Operator Console, which allows human intervention into the AutoSys job stream, and provides various mechanisms for monitoring the AutoSys job stream. Provided with the interface are capabilities to view job definitions, change the state of a job, modify the job definition, and alter job dependency information. Also provided with the Operations interface is an alarm manager. The alarm manager can be used to assist in monitoring jobs and to react to fault situations. These alarms can be set through the definition of a job. More details on the underlying components of AutoSys software are contained in Section 4.6, CSCI Structure. More information about the AutoSys Operator Console is contained in Section 4.7.2, Operator Interfaces.

4.1.4.2 AutoSys Integration into the Processing CSCI Detailed Design

For the Processing CSCI, AutoSys provides the job scheduling engine for the Processing CSCI. AutoSys' Event Processor will manage all the events that occur in the science data production environment. AutoSys' Database will become part of the PDPS Database using the AutoSys provided table schema. For detailed design, the current assumption is that the AutoSys provided processes will manage the processing environment when the production facility is operating in a steady state manor. For start up and shutdown, some development code is needed to assist in establishing communication connections, initializing the PDPS database, assuring the availability of other entities, i.e., Data Server, and alerting operations and other applications about startup and initialization problems. After the completion of startup and upon the completion of adding the daily job schedule to AutoSys, AutoSys will begin managing and monitoring the execution of jobs.

To support the execution of jobs, AutoSys will require additional help in the following areas:

a. **Resource Management**—Allocation of sufficient resources, i.e., disk storage, to support execution. Currently, AutoSys provides no mechanisms for managing disk storage. This is a potential enhancement that will be added to their product. Also, the monitoring of resources will not be done by AutoSys. This is a MSS and Processing joint responsibility.

b. **Data Management**—Manages the staging, destaging, and retention of data on Processing resources.

c. **PGE Execution Management**—Initiates and monitors execution of a PGE.

The following paragraphs briefly illustrates the operational concepts of the new Processing CSCI design. They discuss the following:

a. How a job schedule is created

b. How jobs are prepared and initiated, and

c. How post-processing takes place.

These applications will be initiated by AutoSys at the Job level. For each PGE, a series of preparation, execution, and post-execution jobs will be defined. The preparation jobs will manage the staging of data (if necessary) and allocation of resources to support execution. The execution job will be used to initiate and monitor the execution of the PGE. The post-execution job will be used to destage and deallocate resources.

To accomplish the set-up of these jobs, AutoSys' capabilities to create and manage job dependencies and to create job boxes, which consist of a series of related jobs, will be used. For each Data Processing Request, which contains the data required to support the execution of one PGE, received from Planning, a corresponding job box, which contains a series of jobs to allocate resources, stage data, execute the PGE, destage data, and deallocate resources, will be created. In Figure 4.1-1, the diagram shows the steps involved in providing job information to AutoSys. The Production Planning Workbench component of the Planning CSCI is the initiator of this activity. The Production Planning Workbench component will use the Scheduler (DpPrScheduler) public class which is being created to provide an interface to AutoSys. This class will encapsulate AutoSys defined capabilities and will manage the information flow from Planning to AutoSys. Through the use of API and command line interfaces, the Scheduler class will provide job definitions to the AutoSys Database. Also, through the DpPrScheduler, the Planning CSCI will be able to request the status for the jobs associated with a Data Processing Request.

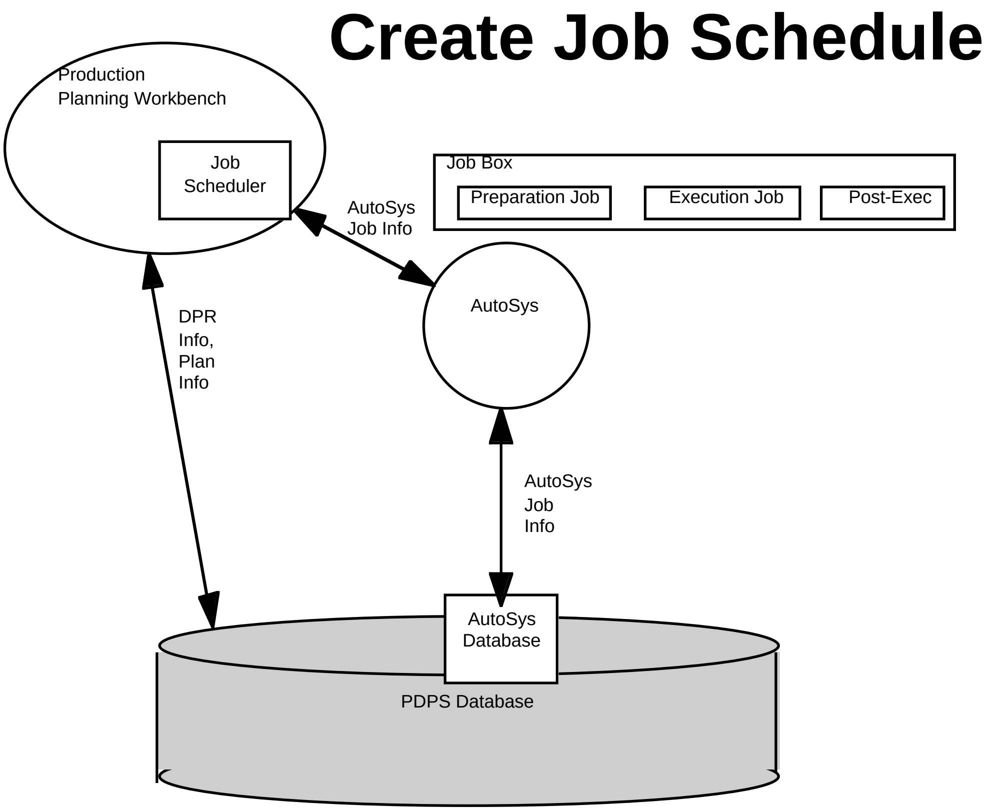


Figure 4.1-1. Scheduling Jobs using AutoSys

Each of these jobs will actually consist of a command which initiates a process or processes to perform the task desired. The process will perform the desired functions and gracefully terminate. The successful completion of their task will result in the next dependent job being released until all jobs within the job box have completed. Besides managing normal operations through these mechanisms, failures which occur within a job box could result in the execution of special fault recovery jobs when necessary.

The Processing CSCI custom components will be initiated by AutoSys to perform support activities, such as resource allocation, staging and destaging data, and interfacing with the PGE. Figure 4.1-2 shows the series of steps and resulting actions performed by the Processing CSCI components.

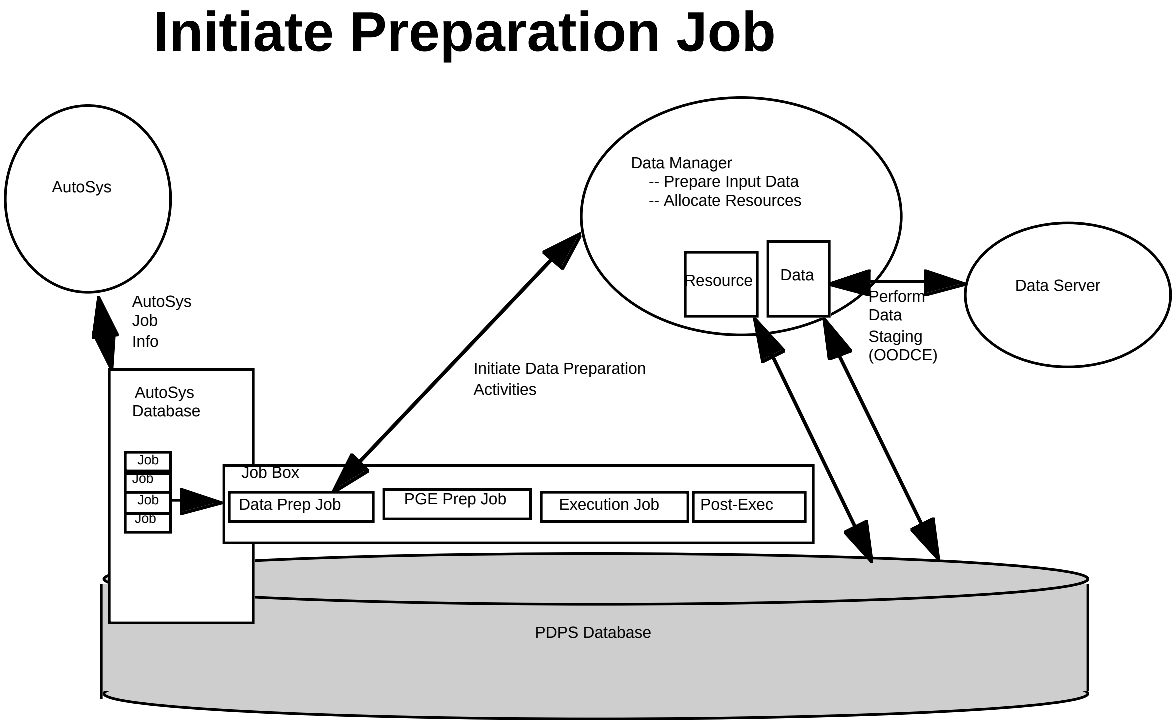


Figure 4.1-2. Initiating Processing Components using AutoSys

AutoSys also provides logging and report generation in order to capture information on job results and status changes. This information will provide information previously mapped to the Processing Log (DpPrProcessingLog) as presented in the Preliminary Design Specification.

4.1.4.3 Platinum Technology's AutoXpert

This product provides mechanisms to monitor and manage the job schedule which currently is being processed in AutoSys. This product is a GUI which provides different methods of viewing the progress of the job schedule, determining corrective measures when required, and providing capabilities to play what-if simulations to determine the affects of modifying the active job schedule. AutoXpert allows the job schedule to be represented at many different levels of abstraction. This concept is an important concept given the large numbers of jobs expected to executed per day at a given DAAC site. These abstract views include a time-line, gantt chart, and job data flows. As part of these views, jobs which are not following predicted behaviors will be color coded which will allow the operations staff to intervene, if necessary. More information about the AutoSys Operator Console and the AutoXpert GUI capabilities is contained in Section 4.6.2, Operator Interfaces.

AutoXpert provides additional capabilities to perform simulations using the current active job schedule. These capabilities will assist the operations staff in judging the downstream effects of the failure of a job, the unavailability of a resource, the late arrival of data, and a job running longer than predicted. These capabilities are provided and can be used in real-time for projections, but also in a simulation mode, where the schedule can be sped up or other job information can be changed to judge what the impact would be. These additional capabilities will help assist DAAC operations personnel in determining what impact production anomalies will have against the active plan and may be used to determine if a replan of production is necessary.

In terms of the PDPS Preliminary Design Specification, AutoXpert provides functions which were previously mapped to the Production Management CSC in the Planning CSCI. Since these functions are being provided by the AutoSys and AutoXpert products, it seemed desirable to map the COTS product to one CSCI. This decision eliminated the need to represent interfaces between AutoSys and AutoXpert across CSCI and Subsystem boundaries.

4.2 CSCI Context

The Processing CSCI interfaces with the following external actors, SDPS and CSMS subsystems to fulfill its responsibilities (see Figure 4.2-1):

a. Planning Subsystem—The Planning Subsystem is responsible for creating a production plan for the Processing CSCI. The Production plan information is conveyed to the Processing CSCI through the use of Data Processing Requests. Each Data Processing Request represents one processing job to be performed by a Data Processing Subsystem computer resource. Processing will provide status information to Planning to assist in production management activities of the Planning CSCI.

b. Data Server Subsystem—To support the creation of ECS Data Products, the Processing CSCI needs the capability of requesting and receiving data (Data Staging) from any of the Data Server resources which has the responsibility of maintaining the data. Also, the Processing CSCI needs the capability of transferring data (Data Destaging) to any Data Server resource for archiving of a generated data product.

c. Operations Interface—To support the management and monitoring of the execution of a PGE and the creation of ECS Data Products, a HMI interface is provided. This interface will provide services to support the collection of status for a Data Processing Request, the cancellation, suspension/resumption and/or modification of a Data Processing Request, and monitoring of the health of Data Processing Subsystem hardware resources. Also, this interface will be used to support manual quality assurance operations performed at the DAAC.

d. SDP Toolkit Interface—To support PGE execution, the Processing CSCI provides information on the location of input data and the location of where the generated output data products are to be maintained. While a PGE is executing, the Processing CSCI monitors the execution of the PGE and informs the operations staff of current status. Status may include current processing event history (what is happening, i.e., data staging, execution). Also, monitoring will be needed to make sure that the processing activity is executing properly. Upon completion of the execution of a PGE, the Data Processing CSCI will inform Planning and will initiate the transfer of the generated data product (if necessary) to the Data Server.

e. CSMS Interface(s)—The Processing CSCI relies on CSMS services to assist in communications and resource management activities and provides system management information to CSMS for Fault, Accounting, Security, Performance, and Accountability purposes. Also, CSMS will provide support for the monitoring of Processing resources and the execution of Science Algorithms.

f. Ingest CSCI—To support science data production, the Processing CSCI must be capable of acquiring necessary input data, the Ingest CSCI's hardware resources are the location of the Level 0 data, and other additional products when first input into ECS.

The Event Flow Diagram (Figure 4.2-2) shows the interactions which occur at the C++ Class Level with Objects which exist in other CSCIs. Table 4.2-2 summarizes the Event Flow.

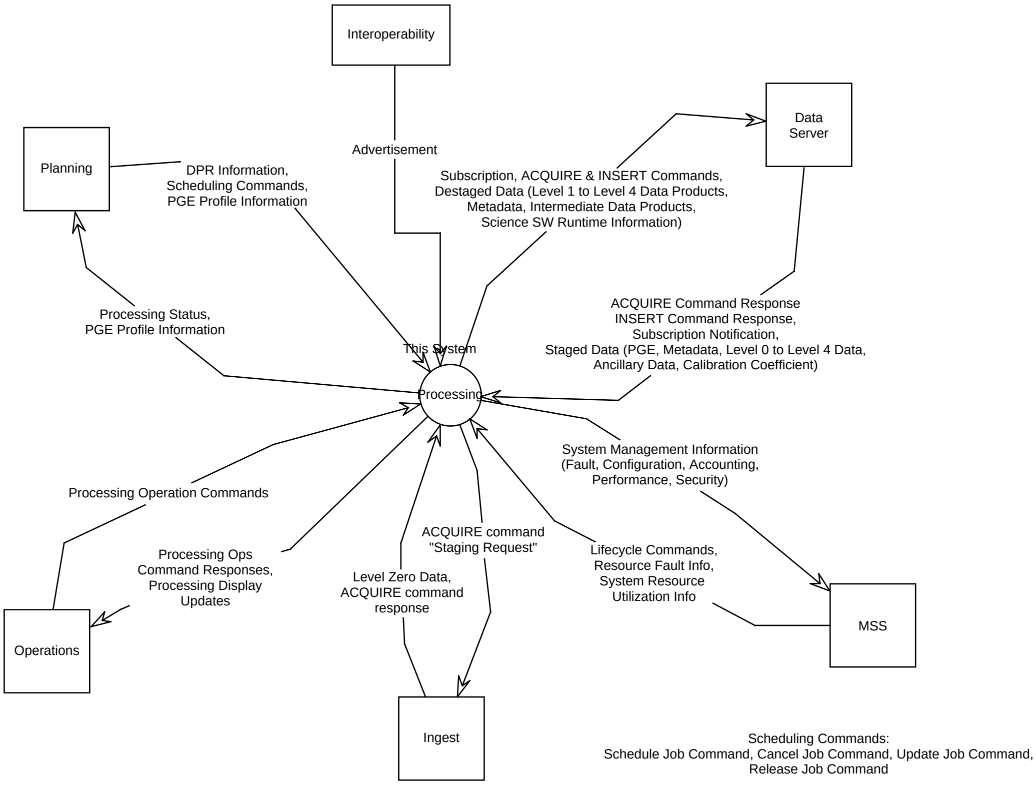


Figure 4.2-1. Processing CSCI Context Diagram

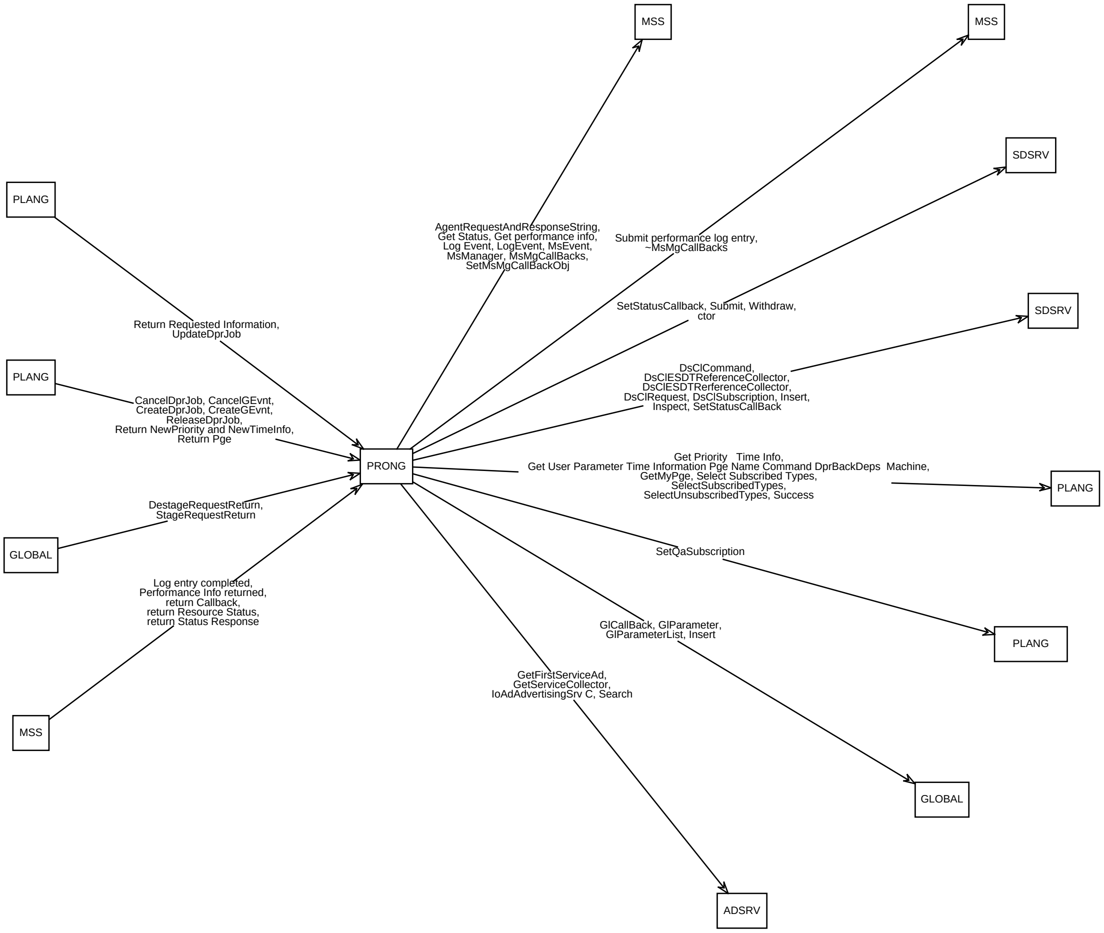

MSS
MSS
SDSRV
PLANG
AgentRequestAndResponseString,
Get Status, Get performance info,
Log Event, LogEvent, MsEvent,
MsManager, MsMgCallBacks,
SetMsMgCallBackObj
Submit performance log entry,
~MsMgCallBacks
SetStatusCallback, Submit, Withdraw,
ctor
SDSRV
Return Requested Information,
UpdateDprJob
PLANG
DsClCommand,
DsClESDTReferenceCollector,
DsClESDTRerferenceCollector,
DsClRequest, DsClSubscription, Insert,
Inspect, SetStatusCallBack
CancelDprJob, CancelGEvnt,
CreateDprJob, CreateGEvnt,
ReleaseDprJob,
Return NewPriority and NewTimeInfo,
Return Pge
PRONG
Get Priority Time Info,
Get User Parameter Time Information Pge Name Command DprBackDeps Machine,
GetMyPge, Select Subscribed Types,
SelectSubscribedTypes,
SelectUnsubscribedTypes, Success
PLANG
DestageRequestReturn,
StageRequestReturn
GLOBAL
SetQaSubscription
Log entry completed,
Performance Info returned,
return Callback,
return Resource Status,
return Status Response
GlCallBack, GlParameter,
GlParameterList, Insert
PLANG
GetFirstServiceAd,
GetServiceCollector,
IoAdAdvertisingSrv C, Search
MSS
GLOBAL
ADSRV

Figure 4.2-2. Processing CSCI Event Flow Summary

Table 4.2-2. PRONG__events Event Flow Summary (1 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	MSS	AgentRequestAnd-ResponseString	
PRONG	MSS	AgentRequestAnd-ResponseString	Status Returned Asynchronously
PRONG	MSS	AgentRequestAnd-ResponseString	Agent Response sent asynchronously
PRONG	MSS	AgentRequestAnd-ResponseString	Agent Response Sent Asynchronously
PLANG	PRONG	CancelDprJob	

Table 4.2-2. PRONG__events Event Flow Summary (2 of 15)

Sender	Receiver	Event Name	Detailed Signature
PLANG	PRONG	CancelGEvnt	
PLANG	PRONG	CreateDprJob	
PLANG	PRONG	CreateGEvnt	
GLOBAL	PRONG	DestageRequestReturn	
GLOBAL	PRONG	DestageRequestReturn	
GLOBAL	PRONG	DestageRequestReturn	Destage Data
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClCommand	
PRONG	SDSRV	DsClESDTReferenceC-ollector	
PRONG	SDSRV	DsClESDTReferenceC-ollector	Create a collector for the dataserver
PRONG	SDSRV	DsClESDTReferenceC-ollector	
PRONG	SDSRV	DsClESDTReferenceC-ollector	

Table 4.2-2. PRONG__events Event Flow Summary (3 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	SDSRV	DsClESDTReferenceC-ollector	
PRONG	SDSRV	DsClESDTReferenceC-ollector	
PRONG	SDSRV	DsClESDTRerferenceC-ollector	Create a collector for the dataserver
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	The constructor places one command in the request
PRONG	SDSRV	DsClRequest	The constructor places one command in the request
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClRequest	
PRONG	SDSRV	DsClSubscription	
PRONG	SDSRV	DsClSubscription	
PRONG	MSS	Get Status	(notify)
PRONG	MSS	Get performance info	(lookup)
PRONG	ADSRV	GetFirstServiceAd	
PRONG	ADSRV	GetFirstServiceAd	

Table 4.2-2. PRONG__events Event Flow Summary (4 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	ADSRV	GetFirstServiceAd	
PRONG	PLANG	GetMyPge	
PRONG	PLANG	GetMyPge	
PRONG	ADSRV	GetServiceCollector	
PRONG	ADSRV	GetServiceCollector	
PRONG	ADSRV	GetServiceCollector	
PRONG	GLOBAL	GlCallBack	ctor(callbackFnPtr)
PRONG	GLOBAL	GlCallBack	
PRONG	GLOBAL	GlCallBack	
PRONG	GLOBAL	GlCallBack	
PRONG	GLOBAL	GlCallBack	
PRONG	GLOBAL	GlParameter	
PRONG	GLOBAL	GlParameter	
PRONG	GLOBAL	GlParameterList	
PRONG	GLOBAL	Insert	[Add an entry to DB]
PRONG	GLOBAL	Insert	[Add an entry into DB]
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	[Add an entry to DB]
PRONG	GLOBAL	Insert	[Add an entry to DB]
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	[Add an entry to DB]

Table 4.2-2. PRONG__events Event Flow Summary (5 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	GLOBAL	Insert	End of iterations
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	
PRONG	GLOBAL	Insert	End of Iterations
PRONG	GLOBAL	Insert	End of iterations
PRONG	SDSRV	Insert	[Add an entry to DB]
PRONG	SDSRV	Insert	[Add an entry into DB]
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	[Add an entry to DB]
PRONG	SDSRV	Insert	[Add an entry to DB]
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	[Add an entry to DB]
PRONG	SDSRV	Insert	End of iterations

Table 4.2-2. PRONG__events Event Flow Summary (6 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	
PRONG	SDSRV	Insert	End of Iterations
PRONG	SDSRV	Insert	End of iterations
PRONG	SDSRV	Inspect	
PRONG	SDSRV	Inspect	
PRONG	SDSRV	Inspect	
PRONG	MSS	Log Event	
MSS	PRONG	Log entry completed	
PRONG	MSS	LogEvent	
PRONG	MSS	LogEvent	
PRONG	MSS	LogEvent	Log event for PGE return condition
PRONG	MSS	LogEvent	Communication Fault
PRONG	MSS	LogEvent	
PRONG	MSS	LogEvent	
PRONG	MSS	MsEvent	
PRONG	MSS	MsEvent	
PRONG	MSS	MsEvent	
PRONG	MSS	MsEvent	
PRONG	MSS	MsEvent	
PRONG	MSS	MsEvent	

Table 4.2-2. PRONG__events Event Flow Summary (7 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	MSS	MsEvent	
PRONG	MSS	MsManager	
PRONG	MSS	MsManager	
PRONG	MSS	MsManager	
PRONG	MSS	MsManager	
PRONG	MSS	MsMgCallBacks	
PRONG	MSS	MsMgCallBacks	
PRONG	MSS	MsMgCallBacks	
PRONG	MSS	MsMgCallBacks	
MSS	PRONG	Performance Info returned	
MSS	PRONG	Performance Info returned	
PLANG	PRONG	ReleaseDprJob	
PLANG	PRONG	Return NewPriority and NewTimeInfo	
PLANG	PRONG	Return Pge	
PLANG	PRONG	Return Requested Information	
PRONG	ADSRV	Search	
PRONG	ADSRV	Search	
PRONG	ADSRV	Search	
PRONG	PLANG	Select Subscribed Types	
PRONG	PLANG	SelectSubscribedTypes	
PRONG	PLANG	SelectUnsubscribed-Types	
PRONG	MSS	SetMsMgCallBackObj	
PRONG	MSS	SetMsMgCallBackObj	
PRONG	MSS	SetMsMgCallBackObj	

Table 4.2-2. PRONG__events Event Flow Summary (8 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	MSS	SetMsMgCallBackObj	
PRONG	MSS	SetMsMgCallBackObj	
PRONG	PLANG	SetQaSubscription	
PRONG	PLANG	SetQaSubscription	
PRONG	PLANG	SetQaSubscription	
PRONG	SDSRV	SetStatusCallBack	
PRONG	SDSRV	SetStatusCallBack	
PRONG	SDSRV	SetStatusCallback	
PRONG	SDSRV	SetStatusCallback	
PRONG	SDSRV	SetStatusCallback	
PRONG	SDSRV	SetStatusCallback	
PRONG	SDSRV	SetStatusCallback	
GLOBAL	PRONG	StageRequestReturn	
GLOBAL	PRONG	StageRequestReturn	
GLOBAL	PRONG	StageRequestReturn	
GLOBAL	PRONG	StageRequestReturn	
PRONG	SDSRV	Submit	
PRONG	SDSRV	Submit	see DpPr_Submit_Staging/ Destaging_Request_To_ DataServer_Event_ Trace
PRONG	SDSRV	Submit	
PRONG	SDSRV	Submit	Synchronous return. Completion notification returned asynchronously.
PRONG	SDSRV	Submit	Synchronous return. Completion notification returned asynchronously.
PRONG	SDSRV	Submit	
PRONG	SDSRV	Submit	

Table 4.2-2. PRONG__events Event Flow Summary (9 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	SDSRV	Submit	Complete Notification to be returned asynchronously
PRONG	SDSRV	Submit	
PRONG	SDSRV	Submit	
PRONG	SDSRV	Submit	
PRONG	SDSRV	Submit	
PRONG	SDSRV	Submit	
PRONG	SDSRV	Submit	
PRONG	SDSRV	Submit	
PRONG	SDSRV	Submit	Complete notify asynchronous
PRONG	MSS	Submit performance log entry	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	

__Table 4.2-2. PRONG__events Event Flow Summary (10 of 15)__

Sender	Receiver	Event Name	Detailed Signature
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PRONG	PLANG	Success	
PLANG	PRONG	UpdateDprJob	
PRONG	SDSRV	Withdraw	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	

Table 4.2-2. PRONG__events Event Flow Summary (11 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	The constructor places one command in the request
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	Create a collector for the dataserver
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	[iterate for all unused data granules]
PRONG	SDSRV	ctor	The constructor places one command in the request
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	Create a collector for the dataserver
PRONG	SDSRV	ctor	The constructor places one command in the request
PRONG	SDSRV	ctor	

Table 4.2-2. PRONG__events Event Flow Summary (12 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	SDSRV	ctor	Create a collector for the dataserver
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	The constructor places one command in the request
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	Create a collector for the dataserver

Table 4.2-2. PRONG__events Event Flow Summary (13 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	SDSRV	ctor	The constructor places one command in the request
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	Create a collector for the dataserver
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	sequenceNumber:int)
PRONG	SDSRV	ctor	(fileName:string
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	(AcsTime:double
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	

Table 4.2-2. PRONG__events Event Flow Summary (14 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	(fileName:string)
PRONG	SDSRV	ctor	QAParameters)
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	(AcsTime:double
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	(fieId:int)
PRONG	SDSRV	ctor	(boxcarWindowSize:int
PRONG	SDSRV	ctor	(fileNames:List<string>
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	(inputFiles:List<string>
PRONG	SDSRV	ctor	(fileName:string)
PRONG	SDSRV	ctor	(fileNames:List<string>
PRONG	SDSRV	ctor	(boxcarWindowSize:int
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	

Table 4.2-2. PRONG__events Event Flow Summary (15 of 15)

Sender	Receiver	Event Name	Detailed Signature
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	
PRONG	SDSRV	ctor	Create a boxcarWindowSize number of ephemeris records and add to queue.
PRONG	SDSRV	ctor	(boxcarWindowSize:int
PRONG	SDSRV	ctor	
MSS	PRONG	return Callback	
MSS	PRONG	return Resource Status	
MSS	PRONG	return Resource Status	
MSS	PRONG	return Resource Status	
MSS	PRONG	return Status Response	
PRONG	MSS	~MsMgCallBacks	

4.3 CSCI Object Model

The Processing CSCI Object Model is actually composed of a number of differing views of components which compose the Processing CSCI. These components provide different abstract views of the Processing CSCI design to assist in developing an understanding of the design. The following object model views will be presented and the underlying objects will be described:

1. Processing CSCI top-level CSC view
2. COTS Manager
3. PGE Execution Manager
4. Data Manager
5. Resource Manager

6. Q/A Monitor
7. Data Pre-Processing

Besides the above listed Processing CSCI components, the Processing CSCI is dependent on the PDPS Database for the management of Processing CSCI persistent data storage. More information on the PDPS Database is contained in the PDPS Database CSC of the Planning CSCI Design Specification.

4.3.1 Processing CSCI Component View

This view of the Processing CSCI (Figure 4.3-1) represents the various components and the associations of these components of the Processing CSCI.

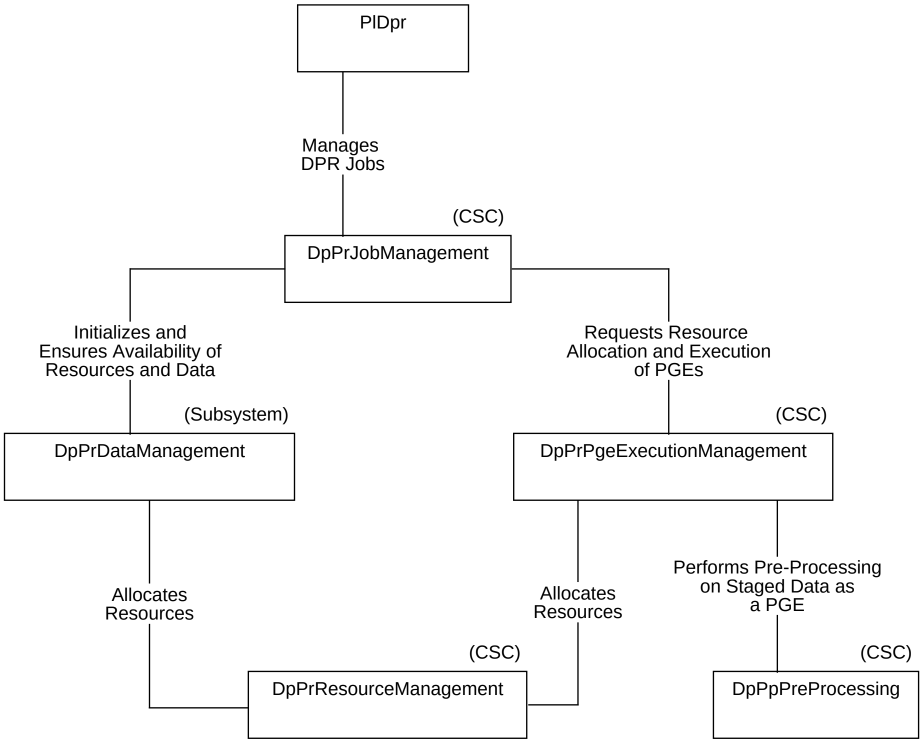

PlDpr
Manages DPR Jobs
(CSC)
DpPrJobManagement
Initializes and Ensures Availability of Resources and Data
Requests Resource Allocation and Execution of PGEs
(Subsystem)
DpPrDataManagement
(CSC)
DpPrPgeExecutionManagement
Allocates Resources
Allocates Resources
Performs Pre-Processing on Staged Data as a PGE
(CSC)
DpPrResourceManagement
(CSC)
DpPpPreProcessing

Figure 4.3-1. Processing CSCI Component View

4.3.2 COTS Manager View

The COTS Manager View (Figure 4.3-2) represents the classes used to manage the flow of information to the COTS products. All ECS applications which need to pass information to the COTS products, which include Planning CSCI components and other Processing CSCI components, will interface to the COTS using these classes. This collection of classes provides the interface to AutoSys, and will take information provided and translate to the COTS supplied API and command line interfaces. Any application which requires an interface to AutoSys would be required to use this interface. The following functions are provided through the COTS Manager:

1. Manages the start-up and shut-down of the AutoSys and AutoXpert products.
2. Provides the interface to AutoSys to add, modify, status, and cancel jobs (Planning CSCI interfaces).
3. Provides additional management/monitoring mechanisms (if needed)
4. Provides the interface to allow the MSS CSCI to interact with AutoSys to alert AutoSys of unavailable resources and external event information, and for AutoSys to provide information to MSS (including Report/log information, if necessary).

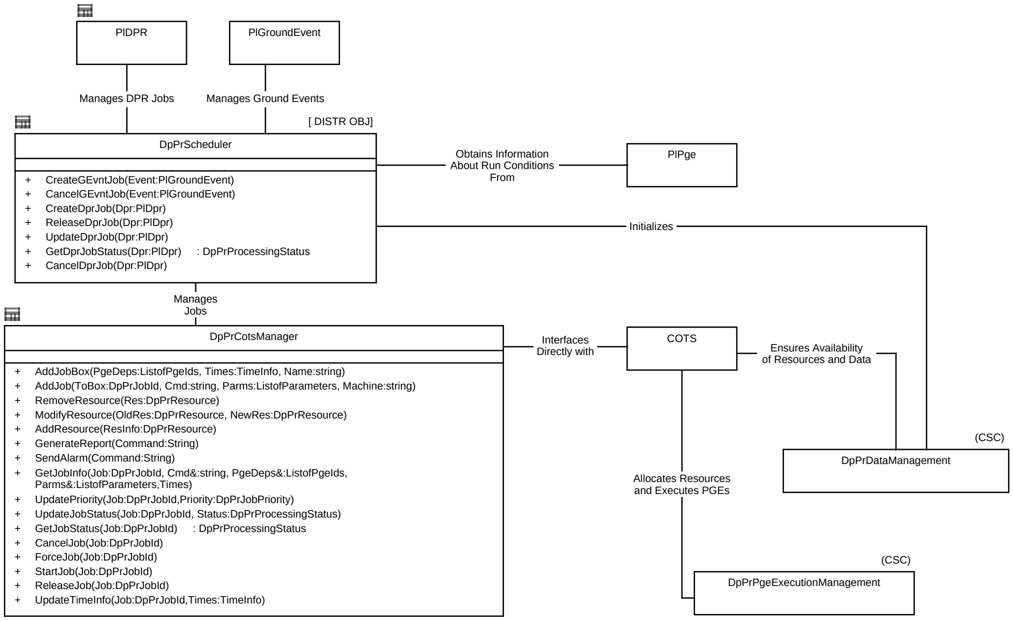

PlDPR
PlGroundEvent
Manages DPR Jobs
Manages Ground Events
[DISTR OBJ]
DpPrScheduler
+ CreateGEvntJob(Event:PlGroundEvent)
+ CancelGEvntJob(Event:PlGroundEvent)
+ CreateDprJob(Dpr:PlDpr)
+ ReleaseDprJob(Dpr:PlDpr)
+ UpdateDprJob(Dpr:PlDpr)
+ GetDprJobStatus(Dpr:PlDpr) : DpPrProcessingStatus
+ CancelDprJob(Dpr:PlDpr)
Obtains Information About Run Conditions From
PlPge
Initializes
Manages Jobs
DpPrCotsManager
+ AddJobBox(PgeDeps:ListofPgeIds, Times:TimeInfo, Name:string)
+ AddJob(ToBox:DpPrJobId, Cmd:string, Parms:ListofParameters, Machine:string)
+ RemoveResource(Res:DpPrResource)
+ ModifyResource(OldRes:DpPrResource, NewRes:DpPrResource)
+ AddResource(ResInfo:DpPrResource)
+ GenerateReport(Command:String)
+ SendAlarm(Command:String)
+ GetJobInfo(Job:DpPrJobId, Cmd&:string, PgeDeps&:ListofPgeIds, Parms&:ListofParameters,Times)
+ UpdatePriority(Job:DpPrJobId,Priority:DpPrJobPriority)
+ UpdateJobStatus(Job:DpPrJobId, Status:DpPrProcessingStatus)
+ GetJobStatus(Job:DpPrJobId) : DpPrProcessingStatus
+ CancelJob(Job:DpPrJobId)
+ ForceJob(Job:DpPrJobId)
+ StartJob(Job:DpPrJobId)
+ ReleaseJob(Job:DpPrJobId)
+ UpdateTimeInfo(Job:DpPrJobId,Times:TimeInfo)
Interfaces Directly with
COTS
Ensures Availability of Resources and Data
(CSC)
DpPrDataManagement
Allocates Resources and Executes PGEs
(CSC)
DpPrPgeExecutionManagement

Figure 4.3-2. COTS Manager View

4.3.3 Data Management View

The Data Management View (Figure 4.3-3) represents the classes used to manage the flow of science data to and from science processing resources. This includes the communication mechanisms needed to interface with the Science Data Server CSCI and the Ingest CSCI. Also, these classes provide additional functions to manage the retention of data on science processing resources which is used to support many PGE executions.

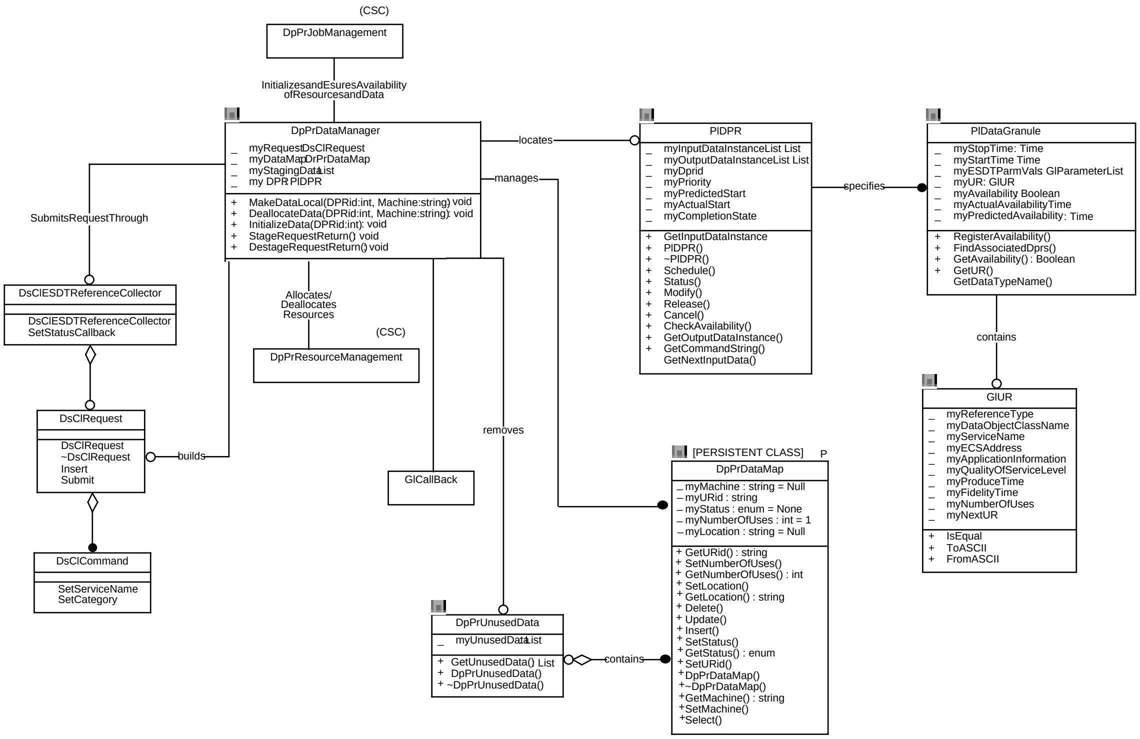


Figure 4.3-3. Data Management View

4.3.4 PGE Execution Management View

The PGE Execution Manager View (Figure 4.3-4) represents the classes used to support the execution of a PGE. While the COTS product will actually initiate the execution of the PGE, the supporting preparation activities, such as creating the Process Control File, are provided through these classes.

4.3.5 Resource Management View

The Resource Management View (Figure 4.3-5) represents the classes used to support the management of science processing resources. These classes provide mappings of logical to the physical resources to allow the Processing CSCI to manage and monitor science processing resources being used to support science data production. This process provides additional resource management and monitoring capabilities that are not currently provided by AutoSys. At this time, it is thought that the interface to Resource Management is through AutoSys, i.e., AutoSys will signal or initiate Resource Management when required. These additional functions will help determine whether all required resources are available to support the execution of the PGE. Please note, that this is one area where Platinum Technology has agreed to strengthen AutoSys' capabilities using PDPS input to determine these additional capabilities.

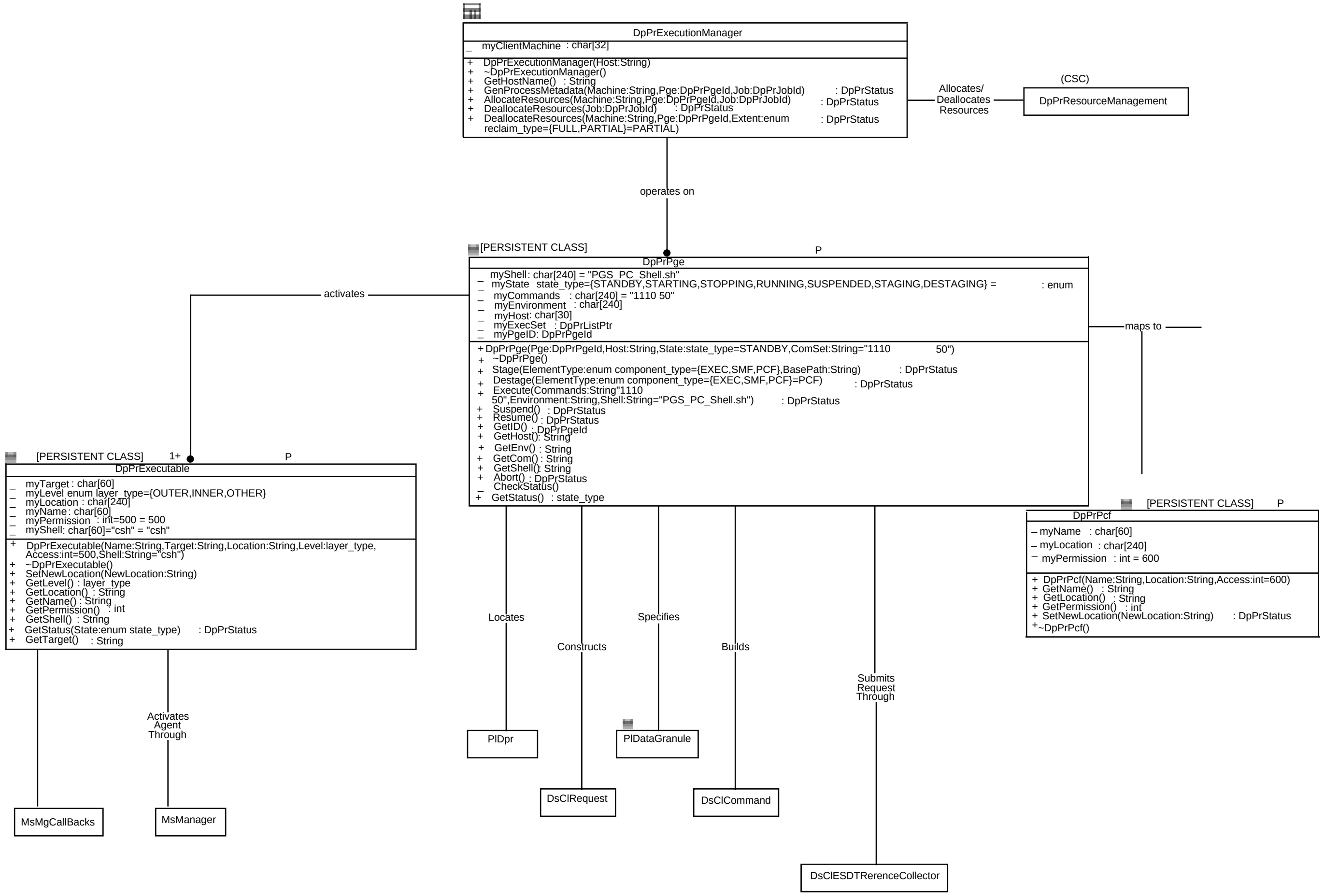

DpPrExecutionManager
_ myClientMachine : char[32]
+ DpPrExecutionManager(Host:String)
+ ~DpPrExecutionManager()
+ GetHostName() : String
+ GenProcessMetadata(Machine:String,Pge:DpPrPgeId,Job:DpPrJobId) : DpPrStatus
+ AllocateResources(Machine:String,Pge:DpPrPgeId,Job:DpPrJobId) : DpPrStatus
+ DeallocateResources(Job:DpPrJobId) : DpPrStatus
+ DeallocateResources(Machine:String,Pge:DpPrPgeId,Extent:enum reclaim_type={FULL,PARTIAL}=PARTIAL) : DpPrStatus
Allocates/ Deallocates Resources
(CSC)
DpPrResourceManagement
operates on
[PERSISTENT CLASS]
P
DpPrPge
_ myShell : char[240] = "PGS_PC_Shell.sh"
_ myState state_type={STANDBY,STARTING,STOPPING,RUNNING,SUSPENDED,STAGING,DESTAGING} = : enum
_ myCommands : char[240] = "1110 50"
_ myEnvironment : char[240]
_ myHost: char[30]
_ myExecSet : DpPrListPtr
_ myPgeID: DpPrPgeId
+ DpPrPge(Pge:DpPrPgeId,Host:String,State:state_type=STANDBY,ComSet:String="1110 50")
+ ~DpPrPge()
+ Stage(ElementType:enum component_type={EXEC,SMF,PCF},BasePath:String) : DpPrStatus
+ Destage(ElementType:enum component_type={EXEC,SMF,PCF}=PCF) : DpPrStatus
+ Execute(Commands:String"1110 50",Environment:String,Shell:String="PGS_PC_Shell.sh") : DpPrStatus
+ Suspend() : DpPrStatus
+ Resume() : DpPrStatus
+ GetID() : DpPrPgeId
+ GetHost(): String
+ GetEnv() : String
+ GetCom() : String
+ GetShell(): String
+ Abort() : DpPrStatus
_ CheckStatus()
+ GetStatus() : state_type
activates
maps to
[PERSISTENT CLASS]
1+
P
DpPrExecutable
_ myTarget : char[60]
_ myLevel enum layer_type={OUTER,INNER,OTHER}
_ myLocation : char[240]
_ myName : char[60]
_ myPermission : int=500 = 500
_ myShell: char[60]="csh" = "csh"
+ DpPrExecutable(Name:String,Target:String,Location:String,Level:layer_type, Access:int=500,Shell:String="csh")
+ ~DpPrExecutable()
+ SetNewLocation(NewLocation:String)
+ GetLevel() : layer_type
+ GetLocation() : String
+ GetName() : String
+ GetPermission() : int
+ GetShell() : String
+ GetStatus(State:enum state_type) : DpPrStatus
+ GetTarget() : String
[PERSISTENT CLASS]
P
DpPrPcf
– myName : char[60]
– myLocation : char[240]
– myPermission : int = 600
+ DpPrPcf(Name:String,Location:String,Access:int=600)
+ GetName() : String
+ GetLocation() : String
+ GetPermission() : int
+ SetNewLocation(NewLocation:String) : DpPrStatus
+~DpPrPcf()
Locates
Constructs
Specifies
Builds
Submits Request Through
PlDpr
PlDataGranule
DsClRequest
DsClCommand
DsClESDTRerenceCollector
Activates Agent Through
MsMgCallBacks
MsManager

Figure 4.3-4. PGE Execution Management View

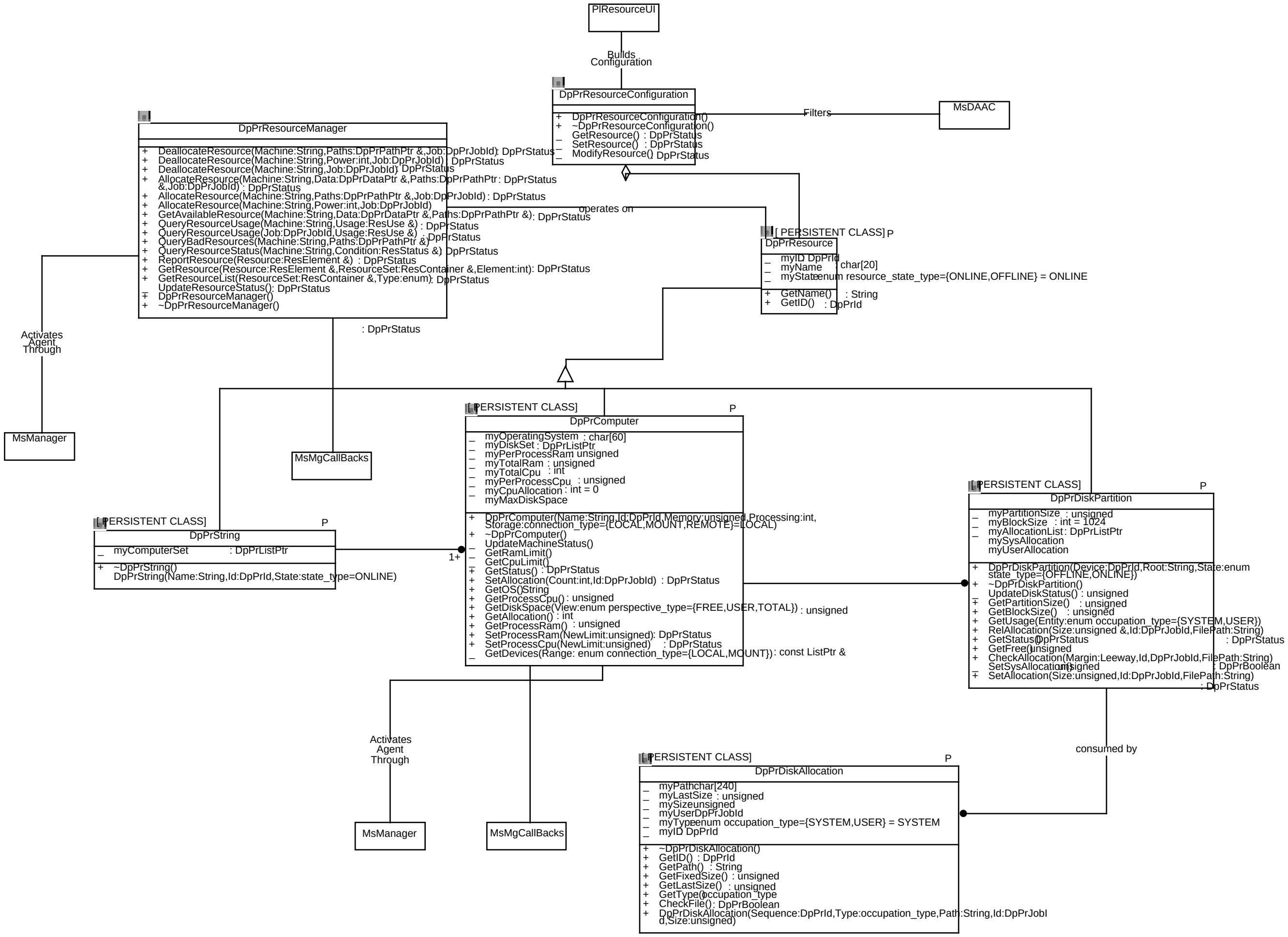

PlResourceUI
Builds Configuration
DpPrResourceConfiguration
+ DpPrResourceConfiguration()
+ ~DpPrResourceConfiguration()
_ GetResource() : DpPrStatus
_ SetResource() : DpPrStatus
_ ModifyResource() : DpPrStatus
Filters
MsDAAC
DpPrResourceManager
+ DeallocateResource(Machine:String,Paths:DpPrPathPtr &,Job:DpPrJobId): DpPrStatus
+ DeallocateResource(Machine:String,Power:int,Job:DpPrJobId) : DpPrStatus
+ DeallocateResource(Machine:String,Job:DpPrJobId): DpPrStatus
+ AllocateResource(Machine:String,Data:DpPrDataPtr &,Paths:DpPrPathPtr : DpPrStatus
&,Job:DpPrJobId) : DpPrStatus
+ AllocateResource(Machine:String,Paths:DpPrPathPtr &,Job:DpPrJobId) : DpPrStatus
+ AllocateResource(Machine:String,Power:int,Job:DpPrJobId)
+ GetAvailableResource(Machine:String,Data:DpPrDataPtr &,Paths:DpPrPathPtr &): DpPrStatus
+ QueryResourceUsage(Machine:String,Usage:ResUse &) : DpPrStatus
+ QueryResourceUsage(Job:DpPrJobId,Usage:ResUse &) : DpPrStatus
+ QueryBadResources(Machine:String,Paths:DpPrPathPtr &) : DpPrStatus
+ QueryResourceStatus(Machine:String,Condition:ResStatus &) : DpPrStatus
+ ReportResource(Resource:ResElement &) : DpPrStatus
+ GetResource(Resource:ResElement &,ResourceSet:ResContainer &,Element:int): DpPrStatus
+ GetResourceList(ResourceSet:ResContainer &,Type:enum) : DpPrStatus
_ UpdateResourceStatus(): DpPrStatus
+ DpPrResourceManager()
+ ~DpPrResourceManager()
: DpPrStatus
operates on
[PERSISTENT CLASS] P
DpPrResource
_ myID DpPrId
_ myName : char[20]
_ myState enum resource_state_type={ONLINE,OFFLINE} = ONLINE
+ GetName() : String
+ GetID() : DpPrId
Activates Agent Through
MsManager
MsMgCallBacks
[PERSISTENT CLASS] P
DpPrString
_ myComputerSet : DpPrListPtr
+ ~DpPrString()
DpPrString(Name:String,Id:DpPrId,State:state_type=ONLINE)
1+
[PERSISTENT CLASS] P
DpPrComputer
_ myOperatingSystem : char[60]
_ myDiskSet : DpPrListPtr
_ myPerProcessRam unsigned
_ myTotalRam : unsigned
_ myTotalCpu : int
_ myPerProcessCpu : unsigned
_ myCpuAllocation : int = 0
_ myMaxDiskSpace
+ DpPrComputer(Name:String,Id:DpPrId,Memory:unsigned,Processing:int, Storage:connection_type={LOCAL,MOUNT,REMOTE}=LOCAL)
+ ~DpPrComputer()
_ UpdateMachineStatus()
_ GetRamLimit()
_ GetCpuLimit()
+ GetStatus() : DpPrStatus
+ SetAllocation(Count:int,Id:DpPrJobId) : DpPrStatus
+ GetOS()String
+ GetProcessCpu() : unsigned
+ GetDiskSpace(View:enum perspective_type={FREE,USER,TOTAL}) : unsigned
+ GetAllocation() : int
+ GetProcessRam() : unsigned
+ SetProcessRam(NewLimit:unsigned): DpPrStatus
+ SetProcessCpu(NewLimit:unsigned) : DpPrStatus
_ GetDevices(Range: enum connection_type={LOCAL,MOUNT}) : const ListPtr &
[PERSISTENT CLASS] P
DpPrDiskPartition
_ myPartitionSize : unsigned
_ myBlockSize : int = 1024
_ myAllocationList : DpPrListPtr
mySysAllocation
myUserAllocation
+ DpPrDiskPartition(Device:DpPrId,Root:String,State:enum state_type={OFFLINE,ONLINE})
+ ~DpPrDiskPartition()
_ UpdateDiskStatus() : unsigned
+ GetPartitionSize() : unsigned
+ GetBlockSize() : unsigned
+ GetUsage(Entity:enum occupation_type={SYSTEM,USER})
+ RelAllocation(Size:unsigned &,Id:DpPrJobId,FilePath:String)
+ GetStatus() DpPrStatus : DpPrStatus
+ GetFree() unsigned
+ CheckAllocation(Margin:Leeway,Id,DpPrJobId,FilePath:String)
_ SetSysAllocation() unsigned : DpPrBoolean
+ SetAllocation(Size:unsigned,Id:DpPrJobId,FilePath:String)
: DpPrStatus
consumed by
Activates Agent Through
MsManager
MsMgCallBacks
[PERSISTENT CLASS] P
DpPrDiskAllocation
_ myPathchar[240]
_ myLastSize : unsigned
_ mySizeunsigned
_ myUserDpPrJobId
_ myType enum occupation_type={SYSTEM,USER} = SYSTEM
_ myID DpPrId
+ ~DpPrDiskAllocation()
+ GetID() : DpPrId
+ GetPath() : String
+ GetFixedSize() : unsigned
+ GetLastSize() : unsigned
+ GetType() occupation_type
+ CheckFile() : DpPrBoolean
+ DpPrDiskAllocation(Sequence:DpPrId,Type:occupation_type,Path:String,Id:DpPrJobId,Size:unsigned)

Figure 4.3-5. Resource Management View

4.3.6 Quality Assurance Monitor View

The Quality Assurance View (Figure 4.3-6) represents the classes used to support the DAAC operations position used to perform DAAC manual quality assurance activities. These activities include visualization of science data products and updating quality assurance metadata.

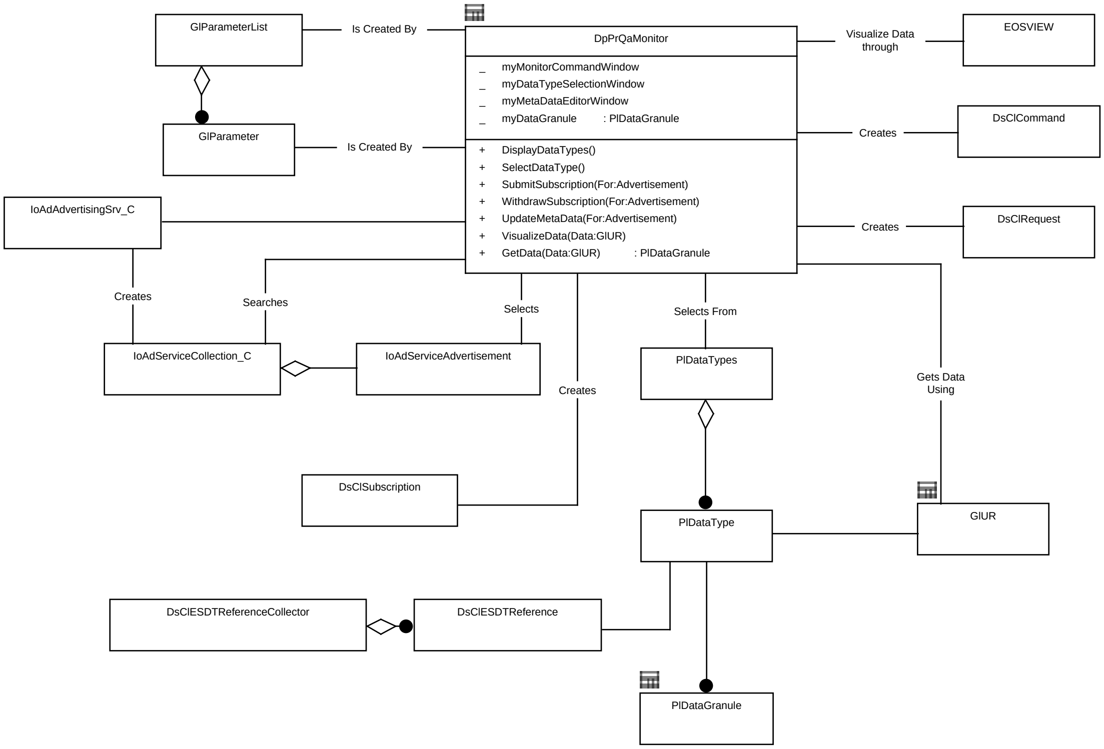


Figure 4.3-6. Quality Assurance Monitor View

4.3.7 Data Pre-Processing View

The Data Pre-Processing View represents the classes used to support the pre-processing of ancillary data which can then be used as inputs to a PGE. Data Preprocessing can be defined as preliminary processing or application of operations on a data set which do not alter or modify its scientific content. Preprocessing includes changes to the format of a data set by reordering the lower level byte structure, reorganization of a data set (ordering data items within and between physical files), preparing additional metadata based on lower level metadata, etc.

Figure 4.3-7, Data Pre-Processing View, represents an overall view of the classes and their underlying associations. For a more detail view, the TRMM Definitive Orbit Object model in Figure 4.3-8 and the TRMM OnBoard Attitude Object model in Figure 4.3-9 have been provided.

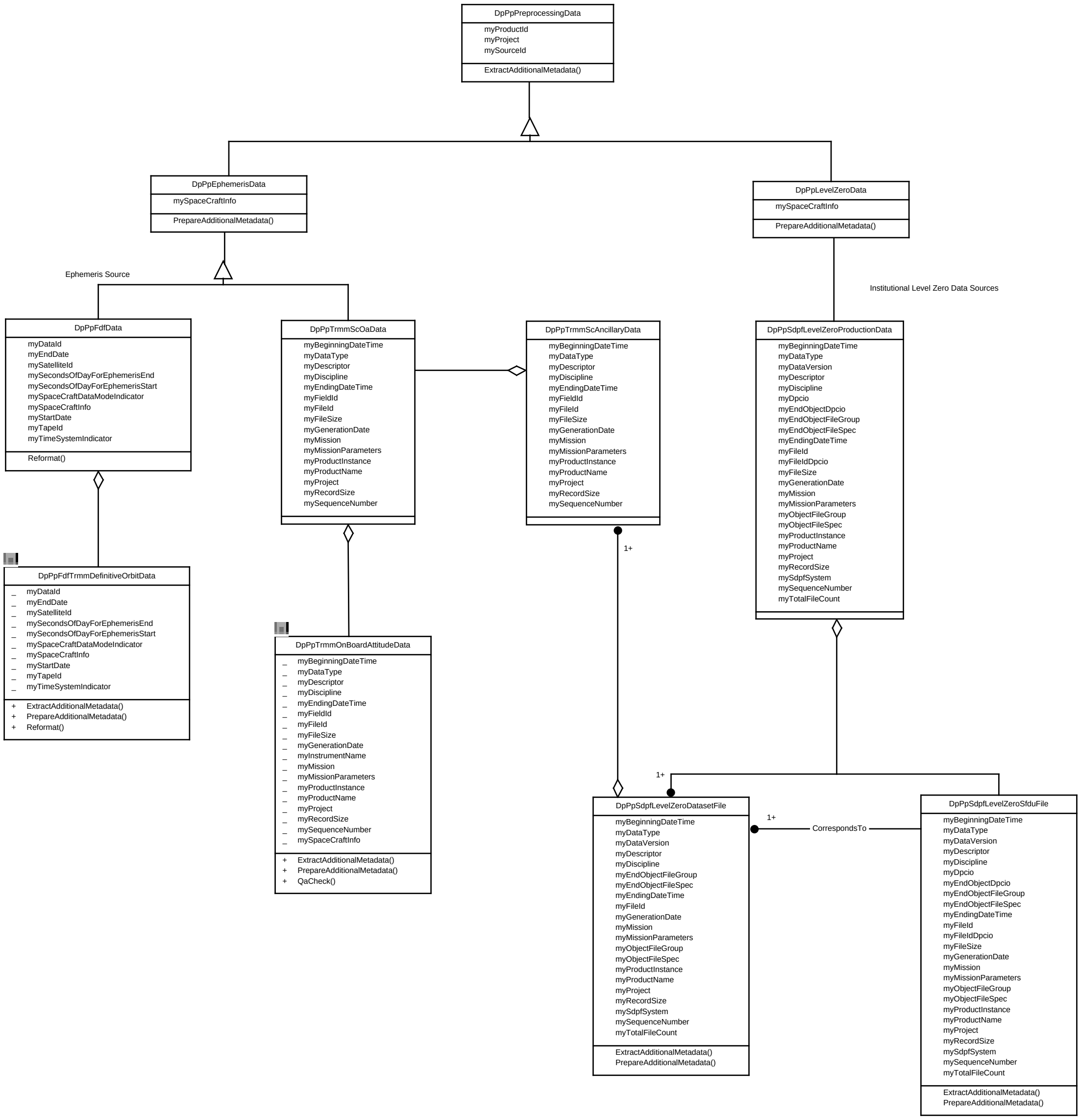


Figure 4.3-7. Data Pre-Processing View

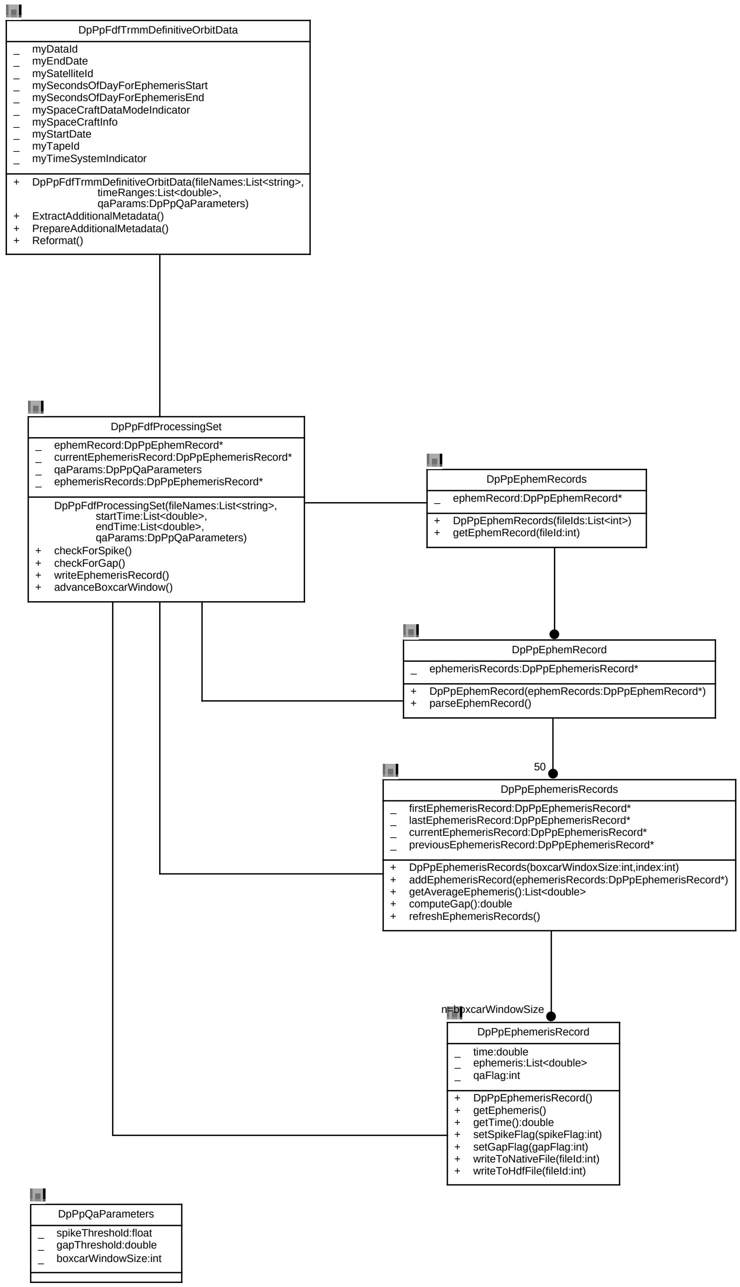

DpPpFdfTrmmDefinitiveOrbitData
_ myDataId
_ myEndDate
_ mySatelliteId
_ mySecondsOfDayForEphemerisStart
_ mySecondsOfDayForEphemerisEnd
_ mySpaceCraftDataModeIndicator
_ mySpaceCraftInfo
_ myStartDate
_ myTapeId
_ myTimeSystemIndicator
+ DpPpFdfTrmmDefinitiveOrbitData(fileNames:List<string>, timeRanges:List<double>, qaParams:DpPpQaParameters)
+ ExtractAdditionalMetadata()
+ PrepareAdditionalMetadata()
+ Reformat()
DpPpFdfProcessingSet
_ ephemRecord:DpPpEphemRecord*
_ currentEphemerisRecord:DpPpEphemerisRecord*
_ qaParams:DpPpQaParameters
_ ephemerisRecords:DpPpEphemerisRecord*
DpPpFdfProcessingSet(fileNames:List<string>, startTime:List<double>, endTime:List<double>, qaParams:DpPpQaParameters)
+ checkForSpike()
+ checkForGap()
+ writeEphemerisRecord()
+ advanceBoxcarWindow()
DpPpEphemRecords
_ ephemRecord:DpPpEphemRecord*
+ DpPpEphemRecords(fileIds:List<int>)
+ getEphemRecord(fileId:int)
DpPpEphemRecord
_ ephemerisRecords:DpPpEphemerisRecord*
+ DpPpEphemRecord(ephemRecords:DpPpEphemRecord*)
+ parseEphemRecord()
50
DpPpEphemerisRecords
_ firstEphemerisRecord:DpPpEphemerisRecord*
_ lastEphemerisRecord:DpPpEphemerisRecord*
_ currentEphemerisRecord:DpPpEphemerisRecord*
_ previousEphemerisRecord:DpPpEphemerisRecord*
+ DpPpEphemerisRecords(boxcarWindoxSize:int,index:int)
+ addEphemerisRecord(ephemerisRecords:DpPpEphemerisRecord*)
+ getAverageEphemeris():List<double>
+ computeGap():double
+ refreshEphemerisRecords()
n=boxcarWindowSize
DpPpEphemerisRecord
_ time:double
_ ephemeris:List<double>
_ qaFlag:int
+ DpPpEphemerisRecord()
+ getEphemeris()
+ getTime():double
+ setSpikeFlag(spikeFlag:int)
+ setGapFlag(gapFlag:int)
+ writeToNativeFile(fileId:int)
+ writeToHdfFile(fileId:int)
DpPpQaParameters
_ spikeThreshold:float
_ gapThreshold:double
_ boxcarWindowSize:int

Figure 4.3-8. Data Pre-Processing View (TRMM Definitive Orbit)

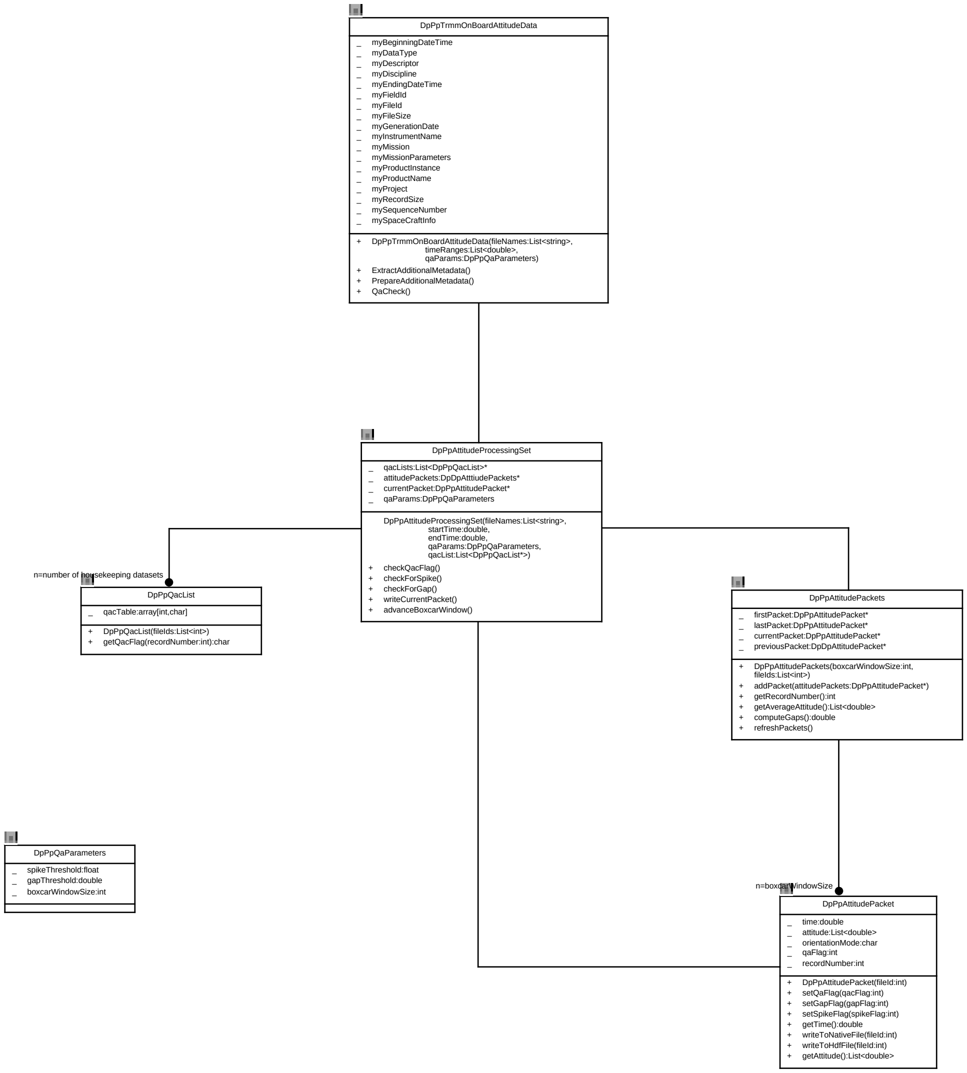

DpPpTrmmOnBoardAttitudeData
_ myBeginningDateTime
_ myDataType
_ myDescriptor
_ myDiscipline
_ myEndingDateTime
_ myFieldId
_ myFileId
_ myFileSize
_ myGenerationDate
_ myInstrumentName
_ myMission
_ myMissionParameters
_ myProductInstance
_ myProductName
_ myProject
_ myRecordSize
_ mySequenceNumber
_ mySpaceCraftInfo
+ DpPpTrmmOnBoardAttitudeData(fileNames:List<string>, timeRanges:List<double>, qaParams:DpPpQaParameters)
+ ExtractAdditionalMetadata()
+ PrepareAdditionalMetadata()
+ QaCheck()
DpPpAttitudeProcessingSet
_ qacLists:List<DpPpQacList>*
_ attitudePackets:DpDpAtttiudePackets*
_ currentPacket:DpPpAttitudePacket*
_ qaParams:DpPpQaParameters
DpPpAttitudeProcessingSet(fileNames:List<string>, startTime:double, endTime:double, qaParams:DpPpQaParameters, qacList:List<DpPpQacList*>)
+ checkQacFlag()
+ checkForSpike()
+ checkForGap()
+ writeCurrentPacket()
+ advanceBoxcarWindow()
n=number of housekeeping datasets
DpPpQacList
_ qacTable:array[int,char]
+ DpPpQacList(fileIds:List<int>)
+ getQacFlag(recordNumber:int):char
DpPpAttitudePackets
_ firstPacket:DpPpAttitudePacket*
_ lastPacket:DpPpAttitudePacket*
_ currentPacket:DpPpAttitudePacket*
_ previousPacket:DpDpAttitudePacket*
+ DpPpAttitudePackets(boxcarWindowSize:int, fileIds:List<int>)
+ addPacket(attitudePackets:DpPpAttitudePacket*)
+ getRecordNumber():int
+ getAverageAttitude():List<double>
+ computeGaps():double
+ refreshPackets()
DpPpQaParameters
_ spikeThreshold:float
_ gapThreshold:double
_ boxcarWindowSize:int
n=boxcarWindowSize
DpPpAttitudePacket
_ time:double
_ attitude:List<double>
_ orientationMode:char
_ qaFlag:int
_ recordNumber:int
+ DpPpAttitudePacket(fileId:int)
+ setQaFlag(qacFlag:int)
+ setGapFlag(gapFlag:int)
+ setSpikeFlag(spikeFlag:int)
+ getTime():double
+ writeToNativeFile(fileId:int)
+ writeToHdfFile(fileId:int)
+ getAttitude():List<double>

Figure 4.3-9. Data Pre-Processing View (TRMM OnBoard Attitude)

4.4 Class Descriptions

The following sections contain descriptions on the classes used in the Processing CSCI components. This information includes descriptions of each class, its attributes, operations and PDL for complex operations. For the Processing CSCI, the COTS product, AutoSys, is not represented through the class descriptions. Information on AutoSys can be found in Section 4.2.4.

4.4.1 COTS Class

The COTS class provides an abstract view of the COTS products and its respective role.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The COTS class has associations with the following classes:

Class: DpPrPgeExecutionManagement AllocatesResourcesandExecutesPGEs
Class: DpPrDataManagement EnsuresAvailabilityofResourcesandData
Class: DpPrCotsManager InterfacesDirectlywith

4.4.2 DpPrComputer Class

Parent Class: DpPrResource

Public: No Distributed Object: No
Persistent Class: True
Purpose and Description:
This class is used to represent the set of computer hardware that is being used for science software processing within the Processing System. All management activities for controlling the use of processing resources are performed by this class.

Attributes:

myCpuAllocation - This attribute defines the number of processors which are currently allocated to the processing of PGEs on this platform. This value is periodically adjusted to account for the allocation and deallocation of processing resources for PGEs This value cannot exceed the total number of CPUs that defined for this platform.
Data Type: int
Privilege: Private
Default Value: 0

myDiskSet - This attribute points to the set of objects which represent the attached storage devices.
Data Type: DpPrListPtr
Privilege: Private
Default Value:

myMaxDiskSpace

myOperatingSystem - This attribute defines the machine type and the current version of the operating system which controls it.
Data Type: char[60]
Privilege: Private
Default Value:

myPerProcessCpu - This attribute defines the per process limit on processing time which may be granted to an individual process. This limit is imposed by the underlying system and may only be increased up to some system defined limit. Any PGE process which exceeds this limit will not run to completion on this platform.
Data Type: unsigned
Privilege: Private
Default Value:

myPerProcessRam - This attribute defines the system defined limit on heap space for a single process. No PGE may activate a process which uses more than the amount defined, without risking the failure of that process.
Data Type: unsigned
Privilege: Private
Default Value:

myTotalCpu - This attribute represents the actual number of individual processors which may be applied to the processing of one or more PGEs. Only individual processors may be allocated to the processing effort for a single PGE.
Data Type: int
Privilege: Private
Default Value:

myTotalRam - The attribute defines the total RAM configuration for the object instance. This value is used as a course gauge when selecting a computing platform, if a specific platform is not chosen, whereas the per process RAM setting is considered to be a hard limit.
Data Type: unsigned
Privilege: Private
Default Value:

Operations:

DpPrComputer - This constructor performs the creation and initialization of the Computer object by establishing the identifiers and configuration settings for the associated Memory and Processor capabilities. The attached storage configuration will also be determined through the creation of disk device objects for the type of connection specified.

Arguments: Name:String,Id:DpPrId,Memory:unsigned,Processing:int, Storage:connection_type={LOCAL,MOUNT,REMOTE}=LOCAL
Return Type: Void
Privilege: Public

GetAllocation - The current processor allocation level is returned to the calling process.
Arguments:
Return Type: int
Privilege: Public

GetCpuLimit - The per process CPU time limit is acquired from the designated platform. This value may be increased by authorized processes.
Arguments:
Return Type: Void
Privilege: Private

GetDevices - Returns a reference to a list of associated Disk Device objects. These objects are constructed and initialized during the process. Either local, or both local and mounted devices are returned based on the designated connection type.
Arguments: Range: enum connection_type={LOCAL,MOUNT}
Return Type: const ListPtr &
Privilege: Private

GetDiskSpace - Depending on the designator used to define the perspective, the requested disk space value is returned to the calling process. The Max Disk Space attribute normally contains the total disk space which can be allocated for user needs and therefore may not be used as the return value unless the designator so indicates.
Arguments: View:enum perspective_type={FREE,USER,TOTAL}
Return Type: unsigned
Privilege: Public

GetOS - The value of the Operating System attribute is returned to the calling process.
Arguments:
Return Type: String
Privilege: Public

GetProcessCpu - The value of the Per Process Cpu attribute is returned to the calling process.
Arguments:
Return Type: unsigned
Privilege: Public

GetProcessRam - The current value of the Per Process Ram attribute is returned to the calling process.
Arguments:

Return Type: unsigned
Privilege: Public

GetRamLimit - The per process Heap limit is obtained from the designated platform This value will be the soft limit unless some authorized process increased the value.
Arguments:
Return Type: Void
Privilege: Private

GetStatus - The state of the designated platform is retrieved and returned to the calling process. The return status is also used to update the object state.
Arguments:
Return Type: DpPrStatus
Privilege: Public

SetAllocation -
Arguments: Count:int,Id:DpPrJobId
Return Type: DpPrStatus
Privilege: Public

SetProcessCpu - The soft limit imposed by the system may be increased up to the predefined hard limit; authorized users may increase this value beyond the hard limit.
Arguments: NewLimit:unsigned
Return Type: DpPrStatus
Privilege: Public

SetProcessRam - The default soft limit, that is imposed by the system, can be increased up to the hard limit for that system; authorized users may increase the value beyond the hard limit.
Arguments: NewLimit:unsigned
Return Type: DpPrStatus
Privilege: Public

UpdateMachineStatus - Acquire the current variable configuration settings for the object and populate those attribute fields. This data consists of the per process CPU and memory settings which may be modified by authorized processes. Also, the current user available disk space is updated.
Arguments:
Return Type: Void
Privilege: Private

~DpPrComputer - This destructor will trigger the deletion of all dependent object instances.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrComputer class has associations with the following classes:
Class: MsManager ActivatesAgentThrough
Class: DpPrDiskPartition
Class: DpPrString
Class: MsMgCallBacks

4.4.3 DpPrCotsManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
DpPrCotsManager is the class which interfaces directly with the scheduling COTS package. This shields the rest of PRONG from knowledge of how to interact with the COTS, and facilitates the possible exchange in the future with a different scheduling package.

Attributes:

None

Operations:

AddJob - This operation takes the input parameters and creates a script which is used to create a Job in the Scheduling COTS. The Job is added to the Job Box specified in the input parameter ToBox.
Arguments: ToBox:DpPrJobId, Cmd:string, Parms:ListofParameters, Machine:string
Return Type: Void
Privilege: Public

AddJobBox - This operation takes data from the input parameters and creates a script which is used to create a Job Box in the Scheduling COTS.
Arguments: PgeDeps:ListofPgeIds, Times:TimeInfo, Name:string
Return Type: Void
Privilege: Public

AddResource - This operation adds the Resource specified in the input parameters to the list of available resources in the Scheduling COTS
Arguments: ResInfo:DpPrResource
Return Type: Void
Privilege: Public

CancelJob - This operation sends a Cancel Job event to the Scheduling COTS to cancel the given Job; if the Job is a Job Box, the enclosed Jobs will also be cancelled.
Arguments: Job:DpPrJobId
Return Type: Void
Privilege: Public

ForceJob - This operation sends a Force event to the Scheduling COTS; the given job will be started even if all its dependencies have not been met.
Arguments: Job:DpPrJobId
Return Type: Void
Privilege: Public

GenerateReport - This operation executes the input command to generate various reports based on the execution statistics collected so far. Status of jobs and resources, historical run-time data, and job definitions ca be generated via this operation.
Arguments: Command:String
Return Type: Void
Privilege: Public

GetJobInfo - This operation takes the JobID as input and passes back (via references) the PGE Dependencies, Parameters, Time Info and execution Command used by that Job.
Arguments: Job:DpPrJobId, Cmd&:string, PgeDeps&:ListofPgeIds, Parms&:ListofParameters,Times
Return Type: Void
Privilege: Public

GetJobStatus - This operation returns the current processing status of the given Job.
Arguments: Job:DpPrJobId
Return Type: DpPrProcessingStatus
Privilege: Public

ModifyResource - This operation removes the Resource specified as OldRes in the input parameters from the list of available resources in the Scheduling COTS, and adds the Resource specified in NewRes.
Arguments: OldRes:DpPrResource, NewRes:DpPrResource
Return Type: Void
Privilege: Public

ReleaseJob - This operation sends an event to the Scheduling COTS to take the given Job out on ON_HOLD status and allow it to run, if all other dependencies are satisfied.
Arguments: Job:DpPrJobId
Return Type: Void
Privilege: Public

RemoveResource - This operation removes the specified Resource from the list of available resources in the Scheduling COTS.
Arguments: Res:DpPrResource
Return Type: Void
Privilege: Public

SendAlarm - While most alarm are generated internally by the Scheduling COTS, this operation allows for specific alarms to be generated by software from outside the COTS package.
Arguments: Command:String
Return Type: Void
Privilege: Public

StartJob - This operation sends a Start Job event to the Scheduling COTS; the Job moves into the STARTING state, but if there are PGE or Data dependencies which have not been satisfied, the Job will WAIT until they are.
Arguments: Job:DpPrJobId
Return Type: Void
Privilege: Public

UpdateJobStatus - This operation replaces the Status of the Job specified with the value specified in the input parameter Status.
Arguments: Job:DpPrJobId, Status:DpPrProcessingStatus
Return Type: Void
Privilege: Public

UpdatePriority - This operation modifies the Priority of the Job specified in the Scheduling COTS to the value specified.
Arguments: Job:DpPrJobId,Priority:DpPrJobPriority
Return Type: Void
Privilege: Public

UpdateTimeInfo - This operation modifies the Time Information (see UpdateDprJob above) in the Database for the given DPR.
Arguments: Job:DpPrJobId,Times:TimeInfo
Return Type: Void
Privilege: Public

Associations:

The DpPrCotsManager class has associations with the following classes:
Class: COTS InterfacesDirectlywith
Class: DpPrScheduler ManagesJobs

4.4.4 DpPrDataManagement Class

The DpPrDataManagement class provides an abstract view of the DataManagement CSC and its respective role.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DpPrDataManagement class has associations with the following classes:

Class: DpPrResourceManagement AllocatesResources
Class: COTS EnsuresAvailabilityofResourcesandData
Class: DpPrScheduler Initializes
Class: DpPrJobManagement InitializesandEnsuresAvailabilityofResourcesandData

4.4.5 DpPrDataManager Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class defines attributes and operations for initializing and managing data granules required by a PGE during its execution. It also ensures data availability before the execution of a PGE. In case of input data is unavailable at our local system disk, it sends request to DataServer to stage data at a specific location that allocated by Resource Management. At the end of the successful execution of a PGE, it asks DataServer to destage (archive) output data and asks Resource Management to deallocate the sources.

Attributes:

my DPR - This attribute contains an address of a DPR.

Data Type: PlDPR
Privilege: Private
Default Value:

myDataMap - A list of DataMap entries.
Data Type: DrPrDataMap
Privilege: Private
Default Value:

myRequest - This attribute contains an address of request that will be sent to DataServer.
Data Type: DsClRequest
Privilege: Private
Default Value:

myStagingData - List of Data_map entries that are being staged at DataServer.
Data Type: List
Privilege: Private
Default Value:

Operations:

DeallocateData - After a PGE is completed, this operation is called (with DPRid) to decrement the NumberOfUses by 1 for each data input granule indicates 1 less PGE uses this data, but it is not necessary to physically delete from the disk until we run out resources. Then it sends request to DataServer to destage (archive) output data.
Arguments: DPRid:int, Machine:string
Return Type: void
Privilege: Public
PDL:{

```
// Iterate through the input data granule list for a particular DPR
// select all input data entries from DataBase that match URid and
// machine and update the NumberOfUses field by decrementing this
// field by 1.

// Iterate through the output data granule list for a particular
// DPR
// Build request and submit request to DataServer to destage output
//  data.
// After output data are successfully destaged,
//  DestageRequestReturn operation is invoked to call
//  ResourceManagement to deallocate resources for all output data.
// Return success code and end process.
```

DestageRequestReturn - This operation is called by CallBack after a destage request was sent to DataServer. If output data is successfully archived, then it goes through all output data entries, deallocate source, and remove them from Data_Map table in DataBase.
Arguments:
Return Type: void

Privilege: Public

```
PDL:{
// Iterate through output data granule list, call
//  ResourceManagement to deallocate resources allocated to each
//  data.
}
```

InitializeData - This public operation is called to initialize data in the Database. When a DPR is first created, this gets called. It goes to each of Data granule and gets a URid. Then it searches through the DataMap table in the Database to find the URid that matches the keyed UR. If it cannot find any entry, then it will create an entry and put it in the DataBase with NumberOfUses field set to 1 (i.e. there is 1 DPR that needs this data granule). As it goes along, if it finds an entry in DataBase that matches the searched URid, then it increments the NumberOfUses field by 1 (i.e. there is one more DPR that needs this data granule).

Arguments: DPRid:int
Return Type: void
Privilege: Public

```
PDL:{
// Iterate through the input data granule list,
// If there is entry in DataBase for this particular data, then
//   increment the NumberOfUses field by 1.
// If no entry returned from DataBase, then add an entry with URid
//   and NumberOfUses = 1 into DataBase.

// At the end of iteration, return success code and end process.
}
```

MakeDataLocal - This operation mainly initializes and ensures availability of data before PGE is executed. First, it ensures that the required data for execution is available at our local disk. If it is not located on our local system disk, it checks to see if this data is already located on a nearby local system disk. If it is, then copy from nearby local disk to this local disk. If it is still not out there anywhere, it then asks the Resource Management to allocate some disk space on local disk and sends request to Data Server to stage data on local disk at the location indicated by Resource Management.

Arguments: DPRid:int, Machine:string
Return Type: void
Privilege: Public

```
PDL:{
// Iterate through the Output Data granules list, and allocate
// resources for all of them.
// If anyone of them are not successfully allocated, return with
// error code and end process.

// Iterate through the Input data granules list, search in DataBase
// for entry that matches the UR for a particular data granule and
// on a particular machine.
// If none of them returned, then search for the same UR but on any nearby machine.
//   If there is an entry returned with nearby location, then copy
//   data from this nearby local disk to this machine.
//   Add this entry with URid and new location into DataBase
//
//   If no entry return still, then
//   Call Resource Management to allocate space for this data
//   granule.
//   Build the request to DataServer to stage data to the location
//   indicated by Resource Management.
//   Add this entry with URid and new location into DataBase with
//   status = STAGING.
// If an entry return from DataBase, then update the NumberOfUses
// by 1.

// After finish iterating through the Input data granules list,
// submit requests that built to DataServer to stage all data at
// one time.
// StageRequestReturn operation is invoked when all staging data
// are successfully staged to set the status of all staging data
// to LOCAL.
// Return with success code and end process.
}
```

StageRequestReturn - This operation is called by CallBack after a stage request was sent to DataServer. If data is successfully staged, then it goes through all staging data entries, updates the Status to LOCAL.

Arguments:

Return Type: void

Privilege: Public

```
PDL:{
// Iterate through staging input data list, update the entry in
// DataBase by setting Status=LOCAL.
}
```

Associations:

The DpPrDataManager class has associations with the following classes:

Class: DpPrResourceManagement Allocates/DeallocatesResources - DpPrDataManager calls DpPrResourceManagement to allocate/deallocate resources for input and output data granules before and after the execution of a PGE.

Class: GlCallBack

Class: DpPrJobManagement InitializesandEnsuresAvailabilityofResourcesandData

Class: DsClESDTReferenceCollector SubmitsRequestThrough - DataManager creates requests to stage/destage data, and it submits those requests through DsClESDTReferenceCollector class.

Class: DsClRequest builds - DpPrDataManger builds request that contains command to stage/destage data at DataServer.

Class: PlDPR locates - DpPrDataManager locates the DPR entry in the DataBase that associated to a provided DPRid.

Class: DpPrDataMap manages - DpPrDataManager manages and manipulates DpPrDataMap objects as entries in Sybase DataBase.

Class: DpPrUnusedData removes - DpPrDataManager physically removes/deletes the data on our local system disk that not being used by any PGE and also deletes the entries in Sybase DataBase associated with them.

4.4.6 DpPrDataMap Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Persistent Class: True

Purpose and Description:

Since this class is a persistent class, it defines attributes and operations for manipulating the instances of objects as the entries in the Sybase DataBase. It can add, delete, update, and select the entry(ies) in the Sybase DataBase.

Attributes:

myLocation - Indicates where, what path on a local machine that this data granule locates on.
Data Type: string
Privilege: Private
Default Value: Null

myMachine - Indicated where, what machine this data granule locates on for a particular DPR.
Data Type: string
Privilege: Private
Default Value: Null

myNumberOfUses - A number indicates how many DPR use(s) this data granule in a day.
Data Type: int
Privilege: Private
Default Value: 1

myStatus - Indicates the status of data that currently stored in DataBase. The possible values are: STAGING, LOCAL, NONE.
Data Type: enum

Privilege: Private
Default Value: None

myURid - Identifier of an UR.
Data Type: string
Privilege: Private
Default Value:

Operations:

Delete - This operation physically deletes a row of DataMap from the DataBase.
Arguments:
Return Type: Void
Privilege: Public

DpPrDataMap - The default constructor for this class.
Arguments:
Return Type: Void
Privilege: Public

GetLocation - A public operation for other class(es) to use to get the Location attribute value.
Arguments:
Return Type: string
Privilege: Public

GetMachine - A public operation for other class(es) to use to get Machine attribute value.
Arguments:
Return Type: string
Privilege: Public

GetNumberOfUses - A public operation for other class(es) to use to get NumberOfUses attribute value.
Arguments:
Return Type: int
Privilege: Public

GetStatus - A public operation for other class(es) to use to get the Status attribute value.
Arguments:
Return Type: enum
Privilege: Public

GetURid - A public operation for other class(es) to use to get URId attribute value.
Arguments:

Return Type: string
Privilege: Public

Insert - This public operation adds a DataMap row or entry into DataBase.
Arguments:
Return Type: Void
Privilege: Public

Select - This public operation deletes a row or entry from Data_map Sybase table.
Arguments:
Return Type: Void
Privilege: Public

SetLocation - A public operation for other class(es) to use to set the Location attribute value.
Arguments:
Return Type: Void
Privilege: Public

SetMachine - A public operation for other class(es) to use to set Machine attribute value.
Arguments:
Return Type: Void
Privilege: Public

SetNumberOfUses - A public operation for other class(es) to use to set NumberOfUses attribute value.
Arguments:
Return Type: Void
Privilege: Public

SetStatus - A public operation for other class(es) to use to set the Status attribute value.
Arguments:
Return Type: Void
Privilege: Public

SetURid - A public operation for other class(es) to use to set the URid attribute value.
Arguments:
Return Type: Void
Privilege: Public

Update - This operation updates a field value of a DataMap row in DataBase.
Arguments:
Return Type: Void
Privilege: Public

~DpPrDataMap - The default destructor for this class.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrDataMap class has associations with the following classes:
Class: DpPrDataManager manages - DpPrDataManager manages and manipulates DpPrDataMap objects as entries in Sybase DataBase.
DpPrUnusedData (Aggregation)

4.4.7 DpPrDiskAllocation Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Persistent Class: True
Purpose and Description:
This class is used to maintain the individual records of disk storage usage which are used to maintain the integrity of storage allocations and deallocations. These records are uniquely associated with a particular disk partition (i.e., file system) and computer.

Attributes:

myID - This is an internal identifier used to uniquely track individual allocations. The User attribute, conversely, is not unique in that there may be many allocations for a single job.
Data Type: DpPrId
Privilege: Private
Default Value:

myLastSize - This value represents the latest size recorded for the file specified by the Path attribute. It will be used to compare against the original allocation size to determine if a file is exceeding its expected size.
Data Type: unsigned
Privilege: Private
Default Value:

myPath - This attribute defines the entire directory path to the file for which this allocation is being made.
Data Type: char[240]
Privilege: Private
Default Value:

mySize - The value represents the size specified for the original allocation request. It will be used to compare against the actual size of the file represented by this allocation to check for an unexpected increase in size.
Data Type: unsigned
Privilege: Private
Default Value:

myType - This attribute defines whether the disk space was allocated for system files or user i.e., science software files.
Data Type: enum occupation_type={SYSTEM,USER}
Privilege: Private
Default Value: SYSTEM

myUser - This attribute holds the value of the identifier specified in the original allocation request and represents the job that is associated with the file defined by the Path attribute.
Data Type: DpPrJobId
Privilege: Private
Default Value:

Operations:

CheckFile - This operation provides the means to check on the existence of the file, associated with this allocation, before actually releasing the allocation.
Arguments:
Return Type: DpPrBoolean
Privilege: Public

DpPrDiskAllocation - This constructor will be used to create the object, as well as define the values of the static attributes ID, Type, User, Size and Path; only the LastSize attribute is considered to be dynamic.
Arguments: Sequence:DpPrId,Type:occupation_type,Path:String,Id:DpPrJobId,Size:unsigned
Return Type: Void
Privilege: Public

GetFixedSize - Retrieve the value of the Size attribute, which maintains the value of the original resource allocation**.**
Arguments:
Return Type: unsigned
Privilege: Public

GetID - Retrieve the unique, sequence generated, identifier which is defined by the attribute ID.
Arguments:

Return Type: DpPrId
Privilege: Public

GetLastSize - This operation retrieves the value of the Last Size attribute.
Arguments:
Return Type: unsigned
Privilege: Public

GetPath - Retrieve the full filepath to the entity defined by this instance of the object.
Arguments:
Return Type: String
Privilege: Public

GetType - This operation identifies this object as being a normal user allocation, or a non-user allocation which implies that the disk space held by the allocation cannot be used directly for science processing purposes.
Arguments:
Return Type: occupation_type
Privilege: Public

~DpPrDiskAllocation - This deallocator will be used to remove the object.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrDiskAllocation class has associations with the following classes:
Class: DpPrDiskPartition consumedby

4.4.8 DpPrDiskPartition Class

Parent Class: DpPrResource
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
This class is used to represent the set of disk storage devices that are being used to contain the input and output data files which are respectively used and produced by the science software, as well as the executable files which comprise the collective set of software known as a PGE. All management activities for controlling the use of the disk resources are performed by this class.

Attributes:

myAllocationList - This pointer identifies the reference to a list of disk resource allocations which are associated with this object.
Data Type: DpPrListPtr
Privilege: Private
Default Value:

myBlockSize - The block size is a fixed value imposed by the particular operating system. It should be used to convert file sizes to units of bytes.
Data Type: int
Privilege: Private
Default Value: 1024

myPartitionSize - This attribute holds the amount of disk space allocated to the file system. This value is fixed during file system creation.
Data Type: unsigned
Privilege: Private
Default Value:

mySysAllocation - This derived attribute will maintain the amount of disk resources which are, by default, allocated to the system on startup. These resources are, in effect, reserved for the duration of the system's activation.

myUserAllocation - This derived attribute maintains the current amount of disk resources which are presently allocated for the use of science processing**.**

Operations:

CheckAllocation - The disposition of a particular allocation is determined by comparing the current use of resources with the amount originally allocated. A margin value will be used to decide if the disposition is good or bad.
Arguments: Margin:Leeway,Id,DpPrJobId,FilePath:String
Return Type: DpPrBoolean
Privilege: Public

DpPrDiskPartition - This contructor will be used to create the object along with its identification and initial state. Following object creation, all associated system allocations will be created and linked to this object. The actual partition size and current allocation amounts will be initialized.
Arguments: Device:DpPrId,Root:String,State:enum state_type={OFFLINE,ONLINE}
Return Type: Void
Privilege: Public

GetBlockSize - The value of the Block Size attribute is returned to the calling process.
Arguments:
Return Type: unsigned
Privilege: Public

GetFree - Compute the total amount of unused space for this partition which is available for immediate allocation.
Arguments:
Return Type: unsigned
Privilege: Public

GetPartitionSize - The value of the Partition Size attribute is returned to the calling process.
Arguments:
Return Type: unsigned
Privilege: Public

GetStatus - The current state of the object is returned to the calling process.
Arguments:
Return Type: DpPrStatus
Privilege: Public

GetUsage - The current allocation amounts are obtained for the specified user type.
Arguments: Entity:enum occupation_type={SYSTEM,USER}
Return Type: unsigned
Privilege: Public

RelAllocation - Individual file allocations are released and the final size of the allocation returned.
Arguments: Size:unsigned &,Id:DpPrJobId,FilePath:String
Return Type: DpPrStatus
Privilege: Public

SetAllocation - Individual resource allocations are generated for science processing uses. The input allocation amount is the reserved amount. The actual amount used by this allocation may be slightly more than the reserved amount to allow for efficient storage.
Arguments: Size:unsigned,Id:DpPrJobId,FilePath:String
Return Type: DpPrStatus
Privilege: Public

SetSysAllocation - During initialization of the object, an initial set of system allocations will be created to account for system and software usage of this partition. This value is assumed to be static for the current configuration of the resources.
Arguments:
Return Type: unsigned

Privilege: Private

UpdateDiskStatus - The current allocation amounts are derived from the associated allocations in order to update the allocation attributes.
Arguments:
Return Type: unsigned
Privilege: Private

~DpPrDiskPartition - This destructor will perform a recursive deletion of all dependent allocation objects.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrDiskPartition class has associations with the following classes:
Class: DpPrComputer
Class: DpPrDiskAllocation consumedby

4.4.9 DpPrExecutable Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Persistent Class: True
Purpose and Description:
This class is used to maintain the state of the science software components which are crucial to support the proper runtime operation of a PGE.

Attributes:

myLevel - The level of the object can be used to determine if it requires direct execution on the part of the Processing System, or if it is indirectly executed from the PGE itself.
Data Type: enum layer_type={OUTER,INNER,OTHER}
Privilege: Private
Default Value:

myLocation - The disk location where the executable resides will be determined initially, but may be modified to reflect a change in resource allocations.
Data Type: char[240]
Privilege: Private
Default Value:

myName - The actual executable or Status Message File (SMF) name is identified by this attribute.
Data Type: char[60]
Privilege: Private
Default Value:

myPermission - The system permission settings may need to be set by the Processing System following the staging of the executable file or Status Message File (SMF). Note that for the latter, the permission should be set to 400.
Data Type: int=500
Privilege: Private
Default Value: 500

myShell - For objects which are shell scripts, as is expected for the main PGE, this shell may need to be explicitly invoked as part of the job command that the COTS Scheduler issues. This attribute does not apply to binary executables and Status Message Files (SMFs).
Data Type: char[60]="csh"
Privilege: Private
Default Value: "csh"

myTarget - This defines the required machine and operating system combination required to execute this process. A null value indicates that the entity is capable of being run on any platform. This value will be used to determine alternate resources for execution if the initially allocated resource ever fails.
Data Type: char[60]
Privilege: Private
Default Value:

Operations:

DpPrExecutable - This constructor will provide for the creation of the object and the initialization of static attributes.
Arguments: Name:String,Target:String,Location:String,Level:layer_type, Access:int=500,Shell:String="csh"
Return Type: Void
Privilege: Public

GetLevel - The Level attribute is returned to the calling process.
Arguments:
Return Type: layer_type
Privilege: Public

GetLocation - The current directory path, as defined by the Location attribute, is returned to the calling process.
Arguments:
Return Type: String
Privilege: Public

GetName - The filename of the object, as defined by the Name attribute, is returned to the calling process.
Arguments:
Return Type: String
Privilege: Public

GetPermission - The system permission settings, as defined by the Permission attribute, are returned to the calling process.
Arguments:
Return Type: int
Privilege: Public

GetShell - The value of the Shell attribute is returned to the calling process.
Arguments:
Return Type: String
Privilege: Public

GetStatus - Retrieve the current last known state of the executable as defined by the State attribute. This value is updated on a periodic basis.
Arguments: State:enum state_type
Return Type: DpPrStatus
Privilege: Public

GetTarget - The Target attribute value is returned to the calling process.
Arguments:
Return Type: String
Privilege: Public

SetNewLocation - To cover the possibility that initially provided disk resources may need to be reallocated, this operation will allow for the update of the objects physical location.
Arguments: NewLocation:String
Return Type: Void
Privilege: Public

~DpPrExecutable - This destructor will perform an orderly cleanup and deletion of the object.
Arguments:
Return Type: Void

Privilege: Public

Associations:

The DpPrExecutable class has associations with the following classes:
Class: MsManager ActivatesAgentThrough
Class: MsMgCallBacks
Class: DpPrPge activates

4.4.10 DpPrExecutionManager Class

Parent Class: Not Applicable
Public: Yes Distributed Object: No
Purpose and Description:
The PrExecutionManager class is the interface class for other classes which require execution services. Such services include the allocation of processing resources and disk resources for the executable files, execution of science software, and deallocation of associated resources.

Attributes:

myClientMachine - This attribute identifies the local machine where this object is running.
Data Type: char[32]
Privilege: Private
Default Value:

Operations:

AllocateResources - The resource allocations required to support the running of a PGE are performed by this operation. Initial runs of a PGE on a specified Machine will trigger the allocation of disk resources required to store the associated executables and runtime Status Message Files (SMFs), and will initiate the staging of these files to the allocated locations. Each activation of this operation will trigger the allocation of processing resources for the specified Job run.
Arguments: Machine:String,Pge:DpPrPgeId,Job:DpPrJobId
Return Type: DpPrStatus
Privilege: Public

DeallocateResources - The processing resources allocated for a particular run of the PGE, as specified by the Job identifier, will be released. All disk resources associated with the PGE remain active.
Arguments: Job:DpPrJobId
Return Type: DpPrStatus

Privilege: Public

DeallocateResources - Processing, and if specified all associated disk resources will be reclaimed for a PGE on the machine indicated.
Arguments: Machine:String,Pge:DpPrPgeId,Extent:enum reclaim_type={FULL,PARTIAL}=PARTIAL
Return Type: DpPrStatus
Privilege: Public

DpPrExecutionManager - This constructor will perform the initialization of the manager object for the platform specified.
Arguments: Host:String
Return Type: Void
Privilege: Public

GenProcessMetadata - This operation will provide for the creation of processing metadata that is associated with the just completed run of a PGE. This metadata will be inserted into the Production History File to provide for the eventual association of this metadata with the Product Output files created by the PGE. This operation will be activated during a post-processing run of the management CSC.
Arguments: Machine:String,Pge:DpPrPgeId,Job:DpPrJobId
Return Type: DpPrStatus
Privilege: Public

GetHostName - The host name of the manager object is returned to the calling process.
Arguments:
Return Type: String
Privilege: Public

~DpPrExecutionManager - This destructor will effect the orderly cleanup and deletion of the manager object.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrExecutionManager class has associations with the following classes:
Class: DpPrResourceManagement Allocates/DeallocatesResources
Class: DpPrPge operateson

4.4.11 DpPrJobManagement Class

The DpPrJobManagement class provides an abstract view of the JobManagement CSC and its respective role.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DpPrJobManagement class has associations with the following classes:
Class: DpPrDataManager InitializesandEnsuresAvailabilityofResourcesandData
Class: DpPrDataManagement InitializesandEnsuresAvailabilityofResourcesandData
Class: PlDpr ManagesDPRJobs
Class: DpPrPgeExecutionManagement RequestsResourceAllocationandExecutionofPGEs

4.4.12 DpPrPcf Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Persistent Class: True
Purpose and Description:
This class is used to maintain the state of the Process Control File (PCF), which provides critical runtime information to the PGE, while the science software is executing. Object instances of this class only persist for the lifetime of a PGE run.

Attributes:

myLocation - This attribute defines the directory path as defined during the initial disk resource allocation.
Data Type: char[240]
Privilege: Private
Default Value:

myName - This is the filename of the Process Control File (PCF) as defined by the Name attribute,
Data Type: char[60]
Privilege: Private

Default Value:

myPermission - This attribute defines the system permissions which need to be placed on the PCF file once it has been stage to the local storage disk.
Data Type: int
Privilege: Private
Default Value: 600

Operations:

DpPrPcf - This constructor provides for the creation of the object and its initialization.
Arguments: Name:String,Location:String,Access:int=600
Return Type: Void
Privilege: Public

GetLocation - Retrieve the directory path of the Process Control File (PCF), as defined by the Location attribute, and return it to the calling process.
Arguments:
Return Type: String
Privilege: Public

GetName - Retrieve the filename of the Process Control File (PCF), as defined by the Name attribute, and return it to the calling process.
Arguments:
Return Type: String
Privilege: Public

GetPermission - Retrieve the system permission settings for the Process Control File (PCF), as defined by the Permission attribute, and return it to the calling process.
Arguments:
Return Type: int
Privilege: Public

SetNewLocation - To cover the possibility that the initially allocated disk resources need to be reallocated, this operation provides for the modification of the Location attribute.
Arguments: NewLocation:String
Return Type: DpPrStatus
Privilege: Public

~DpPrPcf - This destructor will perform an orderly cleanup and deletion of the object.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrPcf class has associations with the following classes:
Class: DpPrPge mapsto

4.4.13 DpPrPge Class

Parent Class: Not Applicable
Public: Yes Distributed Object: No
Persistent Class: True
Purpose and Description:
This class is used to maintain the state of the complete PGE and as such provides for the insertion and deletion of the science software for the local machine and controls execution of the PGE on that platform.

Attributes:

myCommands - The command string will be used to provide startup condition information to the activation shell identified by the Shell attribute. The default command string is specific to the default SDP Toolkit activation shell.
Data Type: char[240]
Privilege: Private
Default Value: "1110 50"

myEnvironment - This attribute may contain the value of zero or more environment variable pairs which will be used to define the operating conditions for the science software.
Data Type: char[240]
Privilege: Private
Default Value:

myExecSet - This pointer identifies a reference to a list of executable files (binaries and shell scripts) as well as Status Message Files (SMFs).
Data Type: DpPrListPtr
Privilege: Private
Default Value:

myHost - This attribute identifies the host machine for this instance of the Pge object. There will be a most one instance of this object per host.
Data Type: char[30]
Privilege: Private
Default Value:

myPgeID - This attribute is used to uniquely identify the science software which is occupying a particular machine. The same identifier may be used as the value of the PgeId

attribute for another instance of this object provided that the value of the Host attribute is different.
Data Type: DpPrPgeId
Privilege: Private
Default Value:

myShell - This attribute defines the Processing shell which activates the science software's outer PGE shell. The default shell is provided by the SDP Toolkit.
Data Type: char[240]
Privilege: Private
Default Value: "PGS_PC_Shell.sh"

myState - The state attribute maintains the last known state of the PGE object. Internal activation of the CheckStatus operation will update this value on a periodic basis.
Data Type: enum state_type={STANDBY,STARTING,STOPPING,RUNNING,SUSPENDED,STAGING, DESTAGING}
Privilege: Private
Default Value: STANDBY

Operations:

Abort - This operation will be used to trigger the premature termination of the currently running science software.
Arguments:
Return Type: DpPrStatus
Privilege: Public

CheckStatus - Used to perform periodic updates of the state information.
Arguments:
Return Type: Void
Privilege: Private

Destage - This operation effectively removes the specified science software files from the local disk.
Arguments: ElementType:enum component_type={EXEC,SMF,PCF}=PCF
Return Type: DpPrStatus
Privilege: Public

DpPrPge - This contructor will be used to create the object instance and initialize the attributes which distinguish it from other instances.
Arguments:
Pge:DpPrPgeId,Host:String,State:state_type=STANDBY,ComSet:String="1110 50"
Return Type: Void

Privilege: Public

Execute - This operation triggers the creation of the Process Control File for the current run of the PGE and updates the COTS Scheduler with the latest job definition information.
Arguments:Commands:String"111050",Environment:String,
Shell:String="PGS_PC_Shell.sh"
Return Type: DpPrStatus
Privilege: Public

GetCom - Retrieves the contents of the Commands attribute and returns it to the calling process.
Arguments:
Return Type: String
Privilege: Public

GetEnv - Retrieves the contents of the Environment attribute and returns it to the calling process.
Arguments:
Return Type: String
Privilege: Public

GetHost - Retrieves the value of the Host attribute and returns it to the calling process.
Arguments:
Return Type: String
Privilege: Public

GetID - Retrieves the value of the PgeId attribute and returns it to the calling process.
Arguments:
Return Type: DpPrPgeId
Privilege: Public

GetShell - Retrieves the name of the activation shell, as defined by the Shell attribute, which will be used by the Processing System to cradle the science software.
Arguments:
Return Type: String
Privilege: Public

GetStatus - Retrieves the state of the object as currently defined by the State attribute.
Arguments:
Return Type: state_type
Privilege: Public

Resume - This operation will trigger the currently suspended science software process to continue with its processing.
Arguments:

Return Type: DpPrStatus
Privilege: Public

Stage - This operation will used to effect the transfer of science software executables and Status Message Files (SMFs). Use of this operation requires that all of the required disk resources be allocated ahead of time.
Arguments: ElementType:enum component_type={EXEC,SMF,PCF},BasePath:String
Return Type: DpPrStatus
Privilege: Public

Suspend - This operation will be activated in order to place the currently executing science software into a suspended state.
Arguments:
Return Type: DpPrStatus
Privilege: Public

~DpPrPge - This destructor will be used to perform an orderly cleanup an deletions of all dependent objects.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrPge class has associations with the following classes:
Class: DsClCommand Builds
Class: DsClRequest Constructs
Class: PlDpr Locates
Class: PlDataGranule Specifies
Class: DsClESDTRerenceCollector SubmitsRequestThrough
Class: DpPrExecutable activates
Class: DpPrPcf mapsto
Class: DpPrExecutionManager operateson

4.4.14 DpPrPgeExecutionManagement Class

The DpPrPgeExecutionManagement class provides an abstract view of the PgeExecutionManagement CSC and its respective role.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DpPrPgeExecutionManagement class has associations with the following classes:
Class: DpPrResourceManagement AllocatesResources
Class: DpPpPreProcessing PerformsPre-ProcessingonStagedDataasaPGE
Class: DpPrJobManagement RequestsResourceAllocationandExecutionofPGEs
Class: COTS AllocatesResourcesandExecutesPGEs

4.4.15 DpPrQaMonitor Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
The DpPrQaMonitor is what the Quality Assurance (QA) position uses to subscribe to and view data products created with QA metadata. The QA operator can enter and withdraw subscriptions to data products, and will be notified via email when the product is available for use. The operator can also modify the QA metadata used to process the subscribed-to product for QA purposes. When the QA position receives email that a subscribed-to product is available, the operator can obtain the product, use tools to visualize the data (based on Data Type) or update the QA metadata used to produce the data.

Attributes:

myDataGranule - This attribute holds the data granule obtained by the GetData or VisualizeData operations.
Data Type: PlDataGranule
Privilege: Private
Default Value:

myDataTypeSelectionWindow - This represents the GUI window to select the data type for subscriptions.
Data Type:
Privilege: Private
Default Value:

myMetaDataEditorWindow - This attribute represents the MetaData Editor which the QA position can use to update the QA MetaData associated with a subscribed-to data product. The UpdateMetaData operation will bring up this Editor, allowing the operator to select a subscribed-to data product. The operator can then change the QA MetaData for use by the GetData or VisualizeData operations.

Data Type:
Privilege: Private
Default Value:

myMonitorCommandWindow - myMonitorCommandWindow represents the main QA Monitor GUI. It will allow the operator to choose the function desired from among subscription submittal and withdrawal, metadata updating, data gathering and data visualization.
Data Type:
Privilege: Private
Default Value:

Operations:

DisplayDataTypes - This operation displays for the operator a list of Data Types which can be subscribed to. The operator can select a Data Type (see SelectDataType operation) and submit or withdraw subscriptions to it.
Arguments:
Return Type: Void
Privilege: Public

GetData - This operation retrieves a product which had been subscribed to by the QA Monitor position.
Arguments: Data:GlUR
Return Type: PlDataGranule
Privilege: Public

SelectDataType - This operation allows the QA operator to select a Data Type from among a list of valid advertised Types. This can then be used in subscription submittal and withdrawal.
Arguments:
Return Type: Void
Privilege: Public

SubmitSubscription - This operation allows the QA operator to subscribe to an advertised Data Type. When a new instance of the Data Type arrives at the Data Server, QA will be sent email notification.
Arguments: For:Advertisement
Return Type: Void
Privilege: Public

UpdateMetaData - This operation is used by the QA Monitor position to change the QA MetaData associated with the given subscribed-to product. This operation will pop-up the MetaData Editor and allow the user to update valid QA MetaData values.

Arguments: For:Advertisement
Return Type: Void
Privilege: Public

```
PDL:{
// The Updated MetaData is inserted into a GlParameter, which is inserted into
//   a GlParameterList.
//
// This ParameterList and the input Advertisement are used to create a
//   DsClCommand, which is then used to create a DsClRequest.
//
// A DsClESDTReferenceCollector is created and used in a Submit call to the
//   previously created DsClRequest. This actually sends the Updated MetaData
//   to the Data Server.
//
}
```

VisualizeData - This operation will be used to produce a visual interpretation of the given data product for QA purposes. Depending on the type of data requested, various tools will be invoked to display these images.
Arguments: Data:GlUR
Return Type: Void
Privilege: Public

WithdrawSubscription - This operation allows the QA operator to "un-subscribe" to an advertised Data Type that was previously subscribed to. New instances of the Data Type arriving at the Data Server will not result in notification.
Arguments: For:Advertisement
Return Type: Void
Privilege: Public

Associations:

The DpPrQaMonitor class has associations with the following classes:
Class: DsClCommand Creates
Class: DsClRequest Creates
Class: DsClSubscription Creates
Class: IoAdAdvertisingSrv_C
Class: GlUR GetsDataUsing
Class: GlParameter IsCreatedBy
Class: GlParameterList IsCreatedBy
Class: IoAdServiceCollection_C Searches
Class: IoAdServiceAdvertisement Selects

Class: PlDataTypes SelectsFrom
Class: EOSVIEW VisualizeDatathrough

4.4.16 DpPrResource Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Persistent Class: True
Purpose and Description:
This base class is used to capture the similar features of the derived classes, and to provide for future expansion.

Attributes:

myID - This base class attribute is inherited by the resource subclasses to uniquely identify object instances.
Data Type: DpPrId
Privilege: Private
Default Value:

myName - This base class attribute is inherited by the resource subclasses to provide an identifier which is more meaningful in human terms.
Data Type: char[20]
Privilege: Private
Default Value:

myState - This base class attribute is inherited by the resource subclasses to define the last known operating state of each object instance.
Data Type: enum resource_state_type={ONLINE,OFFLINE}
Privilege: Private
Default Value: ONLINE

Operations:

GetID - Retrieve the ID attribute for this object instance.
Arguments:
Return Type: DpPrId
Privilege: Public

GetName - Retrieve the Name attribute for this object instance.
Arguments:
Return Type: String

Privilege: Public

Associations:

The DpPrResource class has associations with the following classes:
Class: DpPrResourceManager operateson
DpPrResourceConfiguration (Aggregation)

4.4.17 DpPrResourceConfiguration Class

Parent Class: Not Applicable
Public: Yes Distributed Object: No
Purpose and Description:
This interface class is provided for the sole purpose of filtering hardware configuration information from the CSMS system to the Processing and Planning system.

Attributes:

None

Operations:

DpPrResourceConfiguration - Construct the interface object and create the set of Planning and Processing resource objects from the MSS configuration stores.
Arguments:
Return Type: Void
Privilege: Public

GetResource - Connect to MSS and filter out the subset of processing and planning hardware.
Arguments:
Return Type: DpPrStatus
Privilege: Private

ModifyResource - Update the existing set of hardware resource information to represent the latest configuration for Planning and Processing use.
Arguments:
Return Type: DpPrStatus
Privilege: Private

SetResource - Express the latest set of hardware information in terms of a Planning and Processing configuration.
Arguments:
Return Type: DpPrStatus

Privilege: Private

~DpPrResourceConfiguration - Destroy all resource object instances and then destroy this instance.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrResourceConfiguration class has associations with the following classes:
Class: PlResourceUI BuildsConfiguration
Class: MsDAAC Filters

4.4.18 DpPrResourceManagement Class

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DpPrResourceManagement class has associations with the following classes:
Class: DpPrDataManager Allocates/DeallocatesResources
Class: DpPrDataManagement AllocatesResources
Class: DpPrPgeExecutionManagement AllocatesResources
Class: DpPrExecutionManager Allocates/DeallocatesResources

4.4.19 DpPrResourceManager Class

Parent Class: Not Applicable
Public: Yes Distributed Object: No
Purpose and Description:
This interface class provides an abstract set of operations for effectively managing the collection of processing and storage resources. Proper use of these operations on the part of the Processing System should provide for the same level of resources as were available

to the Planning System during plan generation, thereby helping to ensure that what was planned will be processed.

Attributes:

None

Operations:

AllocateResource - Perform an allocation of disk storage for a set of data items and return the set of allocated storage paths. The allocation is performed for a particular machine and job activation.
Arguments: Machine:String,Data:DpPrDataPtr &,Paths:DpPrPathPtr &,Job:DpPrJobId
Return Type: DpPrStatus
Privilege: Public

AllocateResource - Perform an allocation of disk storage for a predetermined set of storage paths. The allocation is performed for a particular machine and job activation.
Arguments: Machine:String,Paths:DpPrPathPtr &,Job:DpPrJobId
Return Type: DpPrStatus
Privilege: Public

AllocateResource - Perform an allocation of processors for the amount requested. The allocation is performed for a particular machine and job activation.
Arguments: Machine:String,Power:int,Job:DpPrJobId
Return Type: DpPrStatus
Privilege: Public

DeallocateResource - Perform a deallocation of disk storage resources for a particular job, but only for those storage paths indicated.
Arguments: Machine:String,Paths:DpPrPathPtr &,Job:DpPrJobId
Return Type: DpPrStatus
Privilege: Public

DeallocateResource - Perform a deallocation of processing resources for the amount and job indicated.
Arguments: Machine:String,Power:int,Job:DpPrJobId
Return Type: DpPrStatus
Privilege: Public

DeallocateResource - Perform a deallocation of both processing and disk storage resources for a particular job.
Arguments: Machine:String,Job:DpPrJobId
Return Type: DpPrStatus

Privilege: Public

DpPrResourceManager - This constructor will create the object instance for this class and regenerate the resource objects from the persistent database storage.
Arguments:
Return Type: Void
Privilege: Public

GetAvailableResource - For a specific machine, retrieve the set of paths which may be allocated for a particular set of data items; no actual allocation has occurred at this point.
Arguments: Machine:String,Data:DpPrDataPtr &,Paths:DpPrPathPtr &
Return Type: DpPrStatus
Privilege: Public

GetResource - Retrieve information about a particular resource object from the list of objects.
Arguments: Resource:ResElement &,ResourceSet:ResContainer &,Element:int
Return Type: DpPrStatus
Privilege: Public

GetResourceList - Creates a pointer to the set of resource objects specified by the resource type.
Arguments: ResourceSet:ResContainer &,Type:enum
Return Type: DpPrStatus
Privilege: Public

QueryBadResources - Retrieve the set of failed disk allocations for a particular machine.
Arguments: Machine:String,Paths:DpPrPathPtr &
Return Type: DpPrStatus
Privilege: Public

QueryResourceStatus - Retrieve the available resource status information for a particular machine.
Arguments: Machine:String,Condition:ResStatus &
Return Type: DpPrStatus
Privilege: Public

QueryResourceUsage - Determine general resource usage by machine
Arguments: Machine:String,Usage:ResUse &
Return Type: DpPrStatus
Privilege: Public

QueryResourceUsage - Determine general resource usage by job.
Arguments: Job:DpPrJobId,Usage:ResUse &
Return Type: DpPrStatus

Privilege: Public

ReportResource - Generate a summary report containing all available information about a particular resource object.
Arguments: Resource:ResElement &
Return Type: DpPrStatus
Privilege: Public

UpdateResourceStatus - Trigger the status update operation for each resource object to get a current assessment of the entire resource pool.
Arguments:
Return Type: DpPrStatus
Privilege: Private

~DpPrResourceManager - This destructor will update the persistent database storage with the current information contained in the resource objects, then effect the deletion of all resource objects before releasing itself.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrResourceManager class has associations with the following classes:
Class: MsManager ActivatesAgentThrough
Class: MsMgCallBacks
Class: DpPrResource operateson

4.4.20 DpPrScheduler Class

Parent Class: Not Applicable
Public: YesDistributed Object: Yes
Purpose and Description:
DpPrScheduler provides operations to manage science software on a DPR level.

Attributes:

None

Operations:

CancelDprJob - This operation cancels the Job Box and all jobs associated with the input DPR. The Data Manager is contacted to release all resources reserved for the DPR, and to update its data as to the number of DPRs requiring a given data file.

Arguments: Dpr:PlDpr
Return Type: Void
Privilege: Public

CancelGEvntJob - This operation is used by Planning to cancel a previously scheduled Ground Event.
Arguments: Event:PlGroundEvent
Return Type: Void
Privilege: Public

CreateDprJob - This operation is used to convert a Data Processing Request (DPR) into a series of "jobs" to be processed by the Scheduling COTS package. Planning creates a DPR and passes it to the DpPrScheduler via this operation. The information in the DPR is used to create a "Job Box", containing all the steps necessary to successfully run the PGE associated with the DPR. Individual jobs are created for setup of execution resources and for ensuring that all necessary input data is local, as well as running the PGE itself and deallocating the resources and data requirements associated with the DPR. These jobs are entered into the Scheduling COTS software (via the DpPrCotsManager class) to be started when a) All PGE Dependencies have been satisfied, and b) All external data is available at the Data Server.
Arguments: Dpr:PlDpr
Return Type: Void
Privilege: Public

```
PDL:{
// Call the Data Manager to InitializeData, passing it the input DPR ID.
//
// Get the PGE associated with the DPR so that the User Parameters, Time
//   Information, the PGE's name, the Command used to invoke the PGE, the
//   DPRs that this one is dependent upon and the machine on which the PGE
//   is to run, are available.
//
// Call the DpPrCotsManager operation AddJobBox, passing it the DPR
//   dependencies, PGE Name and Time Information. This creates a holder
//   for the various jobs which make up a DPR.
//
// Call the DpPrCotsManager operation AddJob to create a job in the Box which
//   will call the ExecutionManager to Allocate resources for the PGE.
//
// Call the DpPrCotsManager operation AddJob to create a job in the Box which
//   will call the DataManager to MakeDataLocal; all data needed to run the PGE
//   will be placed on local storage.
//
// Call the DpPrCotsManager operation AddJob to create a job in the Box which
//   will call the ExecutionManager to create the PCF and environment files for
//   the PGE; this uses the User Parameters obtained above.
//
// Call the DpPrCotsManager operation AddJob to create a job in the Box which
//   will actually execute the PGE; this uses the Command obtained above.
//
// Call the DpPrCotsManager operation AddJob to create a job in the Box which
//   will call the ExecutionManager to deallocate the resources it had allocated
//   for the PGE.
//
// Call the DpPrCotsManager operation AddJob to create a job in the Box which
//   will call the DataManager to update its information about how many PGEs
//   need to use a certain piece of data. This may result in destaging of data
//   no longer needed by any PGE.
//
}
```

CreateGEvntJob
Arguments: Event:PlGroundEvent
Return Type: Void
Privilege: Public

GetDprJobStatus - This operation returns Processing Status in the Scheduling COTS of the Job Box associated with the DPR. Values can include ON_HOLD, STARTING, WAITING, RUNNING, SUCCESS or FAILURE.
Arguments: Dpr:PlDpr
Return Type: DpPrProcessingStatus
Privilege: Public

ReleaseDprJob - This operation is used by Planning to inform the Scheduler that all input data required from the Data Server is available. At this time, the Scheduler releases the Job Box via the CotsManager, which allows the jobs associated with the DPR to start processing when all the PGE Dependencies have been satisfied (i.i., if all the PGE Dependencies are satisfied but Planning has not called this operation, the DPR cannot processed).
Arguments: Dpr:PlDpr
Return Type: Void
Privilege: Public

UpdateDprJob - This operation is used by Planning to modify a DPR's Priority or Time Information only. Time Information includes minimum and/or maximum start times, end times, and total predicted processing times. These are used for error detection (i.e., if a DPR is taking much longer that predicted to run, an Alarm can be raised). Planning modifies the DPR and passes it to the Scheduler, which contacts the Scheduling COTS (through the CotsManager) to update the information.
Arguments: Dpr:PlDpr
Return Type: Void

Privilege: Public

Associations:

The DpPrScheduler class has associations with the following classes:
Class: DpPrDataManagement Initializes
Class: PlDPR ManagesDPRJobs
Class: PlGroundEvent ManagesGroundEvents
Class: DpPrCotsManager ManagesJobs
Class: PlPge ObtainsInformationAboutRunConditionsFrom

4.4.21 DpPrString Class

Parent Class: DpPrResource
Public: No Distributed Object: No
Persistent Class: True
Purpose and Description:
String is an abstract representation of one or more actual machines

Attributes:

myComputerSet - This pointer attribute references the collection of Computer resources which are associate with this object instance.
Data Type: DpPrListPtr
Privilege: Private
Default Value:

Operations:

DpPrString - Destroy the object instance; this action does not affect the resource object instances which are pointed to by this object.
Arguments: Name:String,Id:DpPrId,State:state_type=ONLINE

~DpPrString - Construct the object instance and define the pointer the collection of Computer resources.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrString class has associations with the following classes:
Class: DpPrComputer

4.4.22 DpPrUnusedData Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is a collection of unused data granules that are candidates to be deleted from our local disk whenever we run out of disk space.

Attributes:

myUnusedData - List of Data_Map entries that not being used (i.e., entries with NumberOfUses = 0) and subject to be deleted from DataBase.
Data Type: List
Privilege: Private
Default Value:

Operations:

DpPrUnusedData
Arguments:
Return Type: Void
Privilege: Public

GetUnusedData - This operation selects all unused entries for a particular machine with NumberOfUses = 0 from DataBase.
Arguments:
Return Type: List
Privilege: Public

~DpPrUnusedData
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpPrUnusedData class has associations with the following classes:
Class: DpPrDataManager removes - DpPrDataManager physically removes/deletes the

data on our local system disk that not being used by any PGE and also deletes the entries in Sybase DataBase associated with them.

4.4.23 DsClCommand Class

This is a Data Server Public Class. Please see Data Server Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DsClCommand class has associations with the following classes:
Class: DpPrPge Builds
DsClRequest (Aggregation)
Class: DpPrQaMonitor Creates

4.4.24 DsClESDTReference Class

This is a Data Server Public Class. Please see Data Server Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DsClESDTReference class has associations with the following classes:
Class: PlDataType

DsClESDTReferenceCollector (Aggregation)

4.4.25 DsClESDTReferenceCollector Class

This is a Data Server Public Class. Please see Data Server Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

DsClESDTReferenceCollector
Arguments:

SetStatusCallback
Arguments:

Associations:

The DsClESDTReferenceCollector class has associations with the following classes:
Class: DpPrDataManager SubmitsRequestThrough
Class: DpPrPge SubmitsRequestThrough

4.4.26 DsClRequest Class

Parent Class: Not Applicable

Attributes:

None

Operations:

DsClRequest
Arguments:

Insert
Arguments:

Submit
Arguments:

~DsClRequest
Arguments:

Associations:

The DsClRequest class has associations with the following classes:
Class: DpPrDataManager builds
DsClESDTReferenceCollector (Aggregation)
Class: DpPrQaMonitor Creates
Class: DpPrPge Constructs

4.4.27 DsClSubscription Class

This is a Data Server Public Class. Please see Data Server Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DsClSubscription class has associations with the following classes:
Class: DpPrQaMonitor Creates

4.4.28 EOSVIEW Class

Please note this is an Abstract class used to represent the EOSVIEW Data visualization tool.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The EOSVIEW class has associations with the following classes:
Class: DpPrQaMonitor VisualizeDatathrough

4.4.29 GlCallBack Class

This is a Global Public Class. Please see CSS Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The GlCallBack class has associations with the following classes:
Class: DpPrDataManager

4.4.30 GlParameter Class

This is a Global Public Class. Please see CSS Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The GlParameter class has associations with the following classes:
Class: DpPrQaMonitor IsCreatedBy
GlParameterList (Aggregation)

4.4.31 GlParameterList Class

This is a Global Public Class. Please see CSS Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The GlParameterList class has associations with the following classes:
Class: DpPrQaMonitor IsCreatedBy

4.4.32 GlUR Class

This is a Global Public Class. Please see CSS Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The GlUR class has associations with the following classes:
Class: PlDataGranule contains
Class: DpPrQaMonitor GetsDataUsing
Class: PlDataType

4.4.33 IoAdAdvertisingSrv_C Class

This is an Interoperability Public Class. Please see Interoperability Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The IoAdAdvertisingSrv_C class has associations with the following classes:
Class: IoAdServiceCollection_C Creates
Class: DpPrQaMonitor

4.4.34 IoAdServiceAdvertisement Class

This is an Interoperability Public Class. Please see Interoperability Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The IoAdServiceAdvertisement class has associations with the following classes:
Class: DpPrQaMonitor Selects
IoAdServiceCollection_C (Aggregation)

4.4.35 IoAdServiceCollection_C Class

This is an Interoperability Public Class. Please see Interoperability Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The IoAdServiceCollection_C class has associations with the following classes:
Class: IoAdAdvertisingSrv_C Creates
Class: DpPrQaMonitor Searches

4.4.36 MsDAAC Class

This is an MSS Public Class. Please see MSS Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The MsDAAC class has associations with the following classes:
Class: DpPrResourceConfiguration Filters

4.4.37 MsManager Class

This is an MSS Public Class. Please see MSS Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The MsManager class has associations with the following classes:
Class: DpPrComputer ActivatesAgentThrough
Class: DpPrExecutable ActivatesAgentThrough
Class: DpPrResourceManager ActivatesAgentThrough

4.4.38 MsMgCallBacks Class

This is an MSS Public Class. Please see MSS Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The MsMgCallBacks class has associations with the following classes:
Class: DpPrComputer

Class: DpPrExecutable
Class: DpPrResourceManager

4.4.39 PlDPR Class

This is a Planning CSCI Public Class. Please see Planning Detailed Design DSpecification for more information.

Parent Class: Not Applicable
Public: Yes Distributed Object: No
Purpose and Description:
This class describes an individual run of a PGE.

Attributes:

None

Operations:

None

Associations:

The PlDPR class has associations with the following classes:
Class: DpPrScheduler ManagesDPRJobs
Class: DpPrDataManager locates
Class: PlDataGranule specifies
Class: DpPrJobManagement ManagesDPRJobs
Class: DpPrPge Locates

4.4.40 PlDataGranule Class

This is a Planning CSCI Public Class. Please see Planning Detailed Design Specification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The PlDataGranule class has associations with the following classes:
Class: PlDataType
Class: GlUR contains
Class: PlDPR specifies
Class: DpPrPge Specifies

4.4.41 PlDataType Class

This is a Planning CSCI Public Class. Please see Planning Detailed Design DSpecification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The PlDataType class has associations with the following classes:
Class: DsClESDTReference
Class: GlUR
Class: PlDataGranule
PlDataTypes (Aggregation)

4.4.42 PlDataTypes Class

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The PlDataTypes class has associations with the following classes:
Class: DpPrQaMonitor SelectsFrom

4.4.43 PlGroundEvent Class

This is a Planning CSCI Public Class. Please see Planning Detailed Design DSpecification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The PlGroundEvent class has associations with the following classes:
Class: DpPrScheduler ManagesGroundEvents

4.4.44 PlPge Class

This is a Planning CSCI Public Class. Please see Planning Detailed Design DSpecification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The PlPge class has associations with the following classes:
Class: DpPrScheduler ObtainsInformationAboutRunConditionsFrom

4.4.45 PlResourceUI Class

This is a Planning CSCI Public Class. Please see Planning Detailed Design DSpecification for more information.

Parent Class: Not Applicable

Attributes:

None

Operations:

4.4.46 DpPpAttitudePacket Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents a single instance of the attitude packet found in the Level Zero Housekeeping dataset.

Attributes:

attitude - The euler angles and rates.
Data Type: double array
Privilege: Private
Default Value:

orientationMode - The spacecraft orientation mode.
Data Type: char
Privilege: Private
Default Value:

qaFlag - The data quality flag for the attitude.
Data Type: int
Privilege: Private
Default Value:

recordNumber - The record number of the attitude packet in the sequence of packets found in the Level Zero Housekeeping dataset.
Data Type: int
Privilege: Private
Default Value:

time - The timestamp of the attitude.
Data Type: double
Privilege: Private
Default Value:

Operations:

DpPpAttitudePacket
Arguments: fileId:int
Return Type: Void
Privilege: Private

getAttitude - Retrieves the attitude from the attitude packet.
Arguments:
Return Type: Void
Privilege: Private

getTime - Retrieves the timestamp from the attitude packet.
Arguments:
Return Type: Void
Privilege: Private

setGapFlag - Sets the gap flag in the attitude data quality summary flag.
Arguments: gapFlag:int
Return Type: Void
Privilege: Private

setQaFlag - Sets the attitude data quality summary flag to the QAC flag.
Arguments: qacFlag:int
Return Type: Void
Privilege: Private

setSpikeFlag - Sets the spike flag in the attitude data quality summary flag.
Arguments: spikeFlag:int
Return Type: Void
Privilege: Private

writeToHdfFile - Appends an attitude packet to the attitude dataset being written in HDF format.
Arguments: fileId:int
Return Type: Void
Privilege: Private

writeToNativeFile - Appends an attitude packet to the attitude dataset being written in the hardware format native to the host machine.
Arguments: fileId:int
Return Type: Void
Privilege: Private

Associations:

The DpPpAttitudePacket class has associations with the following classes:
Class: DpPpAttitudePackets
Class: DpPpAttitudeProcessingSet

4.4.47 DpPpAttitudePackets Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents the set of attitude packets involved in data quality processing.

Attributes:

currentPacket - The current attitude packet undergoing data quality processing.
Data Type: structure
Privilege: Private
Default Value:

firstPacket - The first attitude packet in the data quality processing queue.
Data Type: structure
Privilege: Private
Default Value:

lastPacket - The last attitude packet in the data quality processing queue.
Data Type: structure

Privilege: Private
Default Value:

previousPacket - The attitude packet preceding the current packet in the data quality processing queue.
Data Type: structure
Privilege: Private
Default Value:

Operations:

DpPpAttitudePackets -
Arguments: boxcarWindowSize:int, fileIds:List<int>
Return Type: Void
Privilege: Private

addPacket - Places the initial set of attitude packets in the data quality processing queue. This number is determined by the boxcar averaging window size.
Arguments: attitudePackets:DpPpAttitudePacket*
Return Type: Void
Privilege: Private

computeGaps - Computes the difference in time between the current and previous attitude packets.
Arguments:
Return Type: Void
Privilege: Private

getAverageAttitude - Computes the average attitude from the attitude packets in the data quality processing queue.
Arguments:
Return Type: Void
Privilege: Private

getRecordNumber - Determines the record number of the attitude packet.
Arguments:
Return Type: Void
Privilege: Private

refreshPackets - Advances attitude packets in the data quality processing queue by one in preparation for the next data quality processing sequence.
Arguments:
Return Type: Void

Privilege: Private

Associations:

The DpPpAttitudePackets class has associations with the following classes:
Class: DpPpAttitudePacket
Class: DpPpAttitudeProcessingSet

4.4.48 DpPpAttitudeProcessingSet Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
Coordinates the data quality processing of the attitude data and building of the attitude datasets.

Attributes:

attitudePackets - The set of attitude packets in the data quality processing queue.
Data Type: structure
Privilege: Private
Default Value:

currentPacket - The attitude packet currently being processed; processing includes reformatting, data quality checking and appending to the attitude dataset.
Data Type: structure
Privilege: Private
Default Value:

qaParams - These are the quality assurance parameters that define tolerances for the data quality check. Quantities outside these tolerances are said to have bad quality.
Data Type: structure
Privilege: Private
Default Value:

qacLists - This is a set of pointers to the QAC tables, a pointer to each QAC list retrieved from a Level Zero Housekeeping file.
Data Type: int array
Privilege: Private

Default Value:

Operations:

DpPpAttitudeProcessingSet

Arguments: fileNames:List<string>, startTime:double, endTime:double, qaParams:DpPpQaParameters, qacList:List<DpPpQacList*>

advanceBoxcarWindow - Advances the boxcar averaging window a single attitude packet. This window contains the set of attitude packets used in data quality processing operations.
Arguments:
Return Type: Void
Privilege: Private

checkForGap - Checks for gaps in the attitude timeline.
Arguments:
Return Type: Void
Privilege: Private

checkForSpike - Checks euler angle and rate quality by searching for deviations from the trend.
Arguments:
Return Type: Void
Privilege: Private

checkQacFlag - This retrieves the QAC flag for the specified attitude packet from the appropriate QAC table.
Arguments:
Return Type: Void
Privilege: Private

writeCurrentPacket - Appends an attitude packets to the end of the attitude dataset.
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpAttitudeProcessingSet class has associations with the following classes:
Class: DpPpAttitudePacket
Class: DpPpAttitudePackets

Class: DpPpQacList
Class: DpPpTrmmOnBoardAttitudeData

4.4.49 DpPpEphemRecord Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents the FDF ephemeris record being processed into ephemeris records.

Attributes:

ephemerisRecords - The set of ephemeris data parsed from a single FDF ephemeris record.
Data Type: structure array
Privilege: Private
Default Value:

Operations:

DpPpEphemRecord
Arguments: ephemRecords:DpPpEphemRecord*
Return Type: Void
Privilege: Private

parseEphemRecord - Extracts up to 50 ephemeris records from an FDF ephemeris record.
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpEphemRecord class has associations with the following classes:
Class: DpPpEphemRecords
Class: DpPpEphemerisRecords
Class: DpPpFdfProcessingSet

4.4.50 DpPpEphemRecords Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents the FDF ephemeris records from which the ephemeris dataset is derived.

Attributes:

ephemRecord - The FDF ephemeris record in 'EPHEM' format.
Data Type: structure
Privilege: Private
Default Value:

Operations:

DpPpEphemRecords -
Arguments: fileIds:List<int>
Return Type: Void
Privilege: Private

getEphemRecord - Brings a single FDF ephemeris format record into core for parsing into ephemeris records.
Arguments: fileId:int
Return Type: Void
Privilege: Private

Associations:

The DpPpEphemRecords class has associations with the following classes:
Class: DpPpEphemRecord
Class: DpPpFdfProcessingSet

4.4.51 DpPpEphemerisData Class

Parent Class: DpPpPreprocessingData
Private

Persistent Class:
Purpose and Description:
The ephemeris data is generalized based on its source; the spacecraft ancillary data that is downlinked from the spacecraft (Consultative Committee for Space Data Systems (CCSDS)-formatted and part of the L0 Production Data File for TRMM, and assumed to be CCSDS-formatted and part of the L0 PDS for EOS-AM), and FDF-generated ephemeris products.

Attributes:

mySpaceCraftInfo - Name and information about the spacecraft.
Data Type: structure
Privilege: Private
Default Value:

Operations:

PrepareAdditionalMetadata - The Preprocessing "Prepare" operation prepares metadata for the SDP Toolkit. Metadata that are not explicitly available may be derived from other lower level metadata (e.g., orbit number of O/A data files staged, etc.).
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpEphemerisData class has associations with the following classes:
None

4.4.52 DpPpEphemerisRecord Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents a single instance of an ephemeris record as retrieved from the FDF ephemeris record.

Attributes:

ephemeris - The satellite position and velocity vectors.
Data Type: double array
Privilege: Private
Default Value:

qaFlag - The data quality summary flag for the ephemeris.
Data Type: int
Privilege: Private
Default Value:

time - The timestamp of the ephemeris.
Data Type: double
Privilege: Private
Default Value:

Operations:

DpPpEphemerisRecord
Arguments:
Return Type: Void
Privilege: Private

getEphemeris - Retrieves the ephemeris from the ephemeris record.
Arguments:
Return Type: Void
Privilege: Private

getTime - Retrieves the timestamp from the ephemeris record.
Arguments:
Return Type: Void
Privilege: Private

setGapFlag - Sets the gap flag in the ephemeris data quality summary flag.
Arguments: gapFlag:int
Return Type: Void
Privilege: Private

setSpikeFlag - Sets the spike flag in the ephemeris data quality summary flag.
Arguments: spikeFlag:int
Return Type: Void
Privilege: Private

writeToHdfFile - Appends an ephemeris record to the ephemeris dataset being written in HDF format.
Arguments: fileId:int
Return Type: Void
Privilege: Private

writeToNativeFile - Appends an ephemeris record to the ephemeris dataset being written in the hardware format native to the host machine.
Arguments: fileId:int
Return Type: Void
Privilege: Private

Associations:

The DpPpEphemerisRecord class has associations with the following classes:
Class: DpPpEphemerisRecords
Class: DpPpFdfProcessingSet

4.4.53 DpPpEphemerisRecords Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents the set of ephemeris records involved in data processing, and is a subset of ephemeris records parsed from an FDF ephemeris record.

Attributes:

currentEphemerisRecord - The current ephemeris record undergoing data quality processing.
Data Type: structure
Privilege: Private
Default Value:

firstEphemerisRecord - The first ephemeris record in the data quality processing queue.
Data Type: structure
Privilege: Private
Default Value:

lastEphemerisRecord - The last ephemeris record in the data quality processing queue.
Data Type: structure
Privilege: Private
Default Value:

previousEphemerisRecord - The ephemeris record preceding the current ephemeris record in the data quality processing queue.
Data Type: structure
Privilege: Private
Default Value:

Operations:

DpPpEphemerisRecords
Arguments: boxcarWindoxSize:int,index:int
Return Type: Void
Privilege: Private

addEphemerisRecord - Places the initial set of ephemeris records in the data quality processing queue. This number is determined by the boxcar averaging window size.
Arguments: ephemerisRecords:DpPpEphemerisRecord*
Return Type: Void
Privilege: Private

computeGap - Computes the difference in time between the current and previous ephemeris records maintained in the data quality processing queue.
Arguments:
Return Type: Void
Privilege: Private

getAverageEphemeris - Produces the average ephemeris from those ephemeris records in the data quality processing queue.
Arguments:
Return Type: Void
Privilege: Private

refreshEphemerisRecords - Advances ephemeris records in the data quality processing queue by one in preparation for the next data quality processing sequence.
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpEphemerisRecords class has associations with the following classes:
Class: DpPpEphemRecord
Class: DpPpEphemerisRecord
Class: DpPpFdfProcessingSet

4.4.54 DpPpFdfData Class

Parent Class: DpPpEphemerisData
Private
Persistent Class:
Purpose and Description:
This class represents all ephemeris data products generated by FDF for both TRMM and EOS-AM. For TRMM, the definitive orbit data from FDF comes via SDPF.

Attributes:

myDataId - The data ID is a bit configuration assigned to a particular mission. The FDF assigns the number in the mission unique ICD.
Data Type: char
Privilege: Private
Default Value:

myEndDate - End date of ephemeris file.
Data Type: double
Privilege: Private
Default Value:

mySatelliteId - Identifies the satellite the ephemeris is based on.
Data Type: char
Privilege: Private
Default Value:

mySecondsOfDayForEphemerisEnd - The seconds of day count.
Data Type: double
Privilege: Private
Default Value:

mySecondsOfDayForEphemerisStart - The seconds of day count.
Data Type: double
Privilege: Private

Default Value:

mySpaceCraftDataModeIndicator - The spacecraft data mode indicator is a mission dependent designation of the kind of data. The meaning of the data mode indicator for FDF products will be standardized.
Data Type: char
Privilege: Private
Default Value:

mySpaceCraftInfo - Information about the spacecraft.
Data Type: structure
Privilege: Private
Default Value:

myStartDate - Start date of the ephemeris file.
Data Type: double
Privilege: Private
Default Value:

myTapeId - The tape identifier is always "standard" and is stored as the characters STANDARD.
Data Type: char
Privilege: Private
Default Value:

myTimeSystemIndicator - The time system (atomic time, universal time coordinated (UTC)) used in the ephemeris file.
Data Type: char
Privilege: Private
Default Value:

Operations:

Reformat - The SDP Toolkit ephemeris tools require data to be in an uniform format independent of the source (FDF or spacecraft). The Preprocessing reformat functions perform the needed operations to convert data to a format acceptable to the SDP Toolkit. The data can also be converted to HDF-EOS. The most efficient format to handle ephemeris data is TBD.
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpFdfData class has associations with the following classes:
None

4.4.55 DpPpFdfProcessingSet Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
Coordinates the reduction of FDF EPHEM format records to ephemeris dataset format and the data quality processing of the ephemeris.

Attributes:

currentEphemerisRecord - The ephemeris record currently being processed; processing can include reformatting, data quality checking and appending to the ephemeris dataset.
Data Type: structure
Privilege: Private
Default Value:

ephemRecord - The FDF ephemeris record in 'EPHEM' format.
Data Type: structure
Privilege: Private
Default Value:

ephemerisRecords - The set of ephemeris records in the data quality processing queue.
Data Type: structure array
Privilege: Private
Default Value:

qaParams - These are the quality assurance parameters that define tolerances for the data quality check. Quantities outside these tolerances are said to have bad data quality.
Data Type: structure
Privilege: Private
Default Value:

Operations:

DpPpFdfProcessingSet

Arguments: fileNames:List<string>, startTime:List<double>, endTime:List<double>, qaParams:DpPpQaParameters

advanceBoxcarWindow - Advances the boxcar averaging window a single ephemeris record. This window contains the set of ephemeris records used in data quality processing operations.
Arguments:
Return Type: Void
Privilege: Private

checkForGap - Checks for gaps in the ephemeris timeline.
Arguments:
Return Type: Void
Privilege: Private

checkForSpike - Checks satellite position and velocity vector quality by searching for deviation from the trend.
Arguments:
Return Type: Void
Privilege: Private

writeEphemerisRecord - Appends an ephemeris record to the end of the ephemeris dataset.
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpFdfProcessingSet class has associations with the following classes:
Class: DpPpEphemRecord
Class: DpPpEphemRecords
Class: DpPpEphemerisRecord
Class: DpPpEphemerisRecords
Class: DpPpFdfTrmmDefinitiveOrbitData

4.4.56 DpPpFdfTrmmDefinitiveOrbitData Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents TRMM definitive orbit products from FDF provided by SDPF.

Attributes:

myDataId - The data ID is a bit configuration assigned to a particular mission. The FDF assigns the number in the mission unique ICD.
Data Type: char array
Privilege: Private
Default Value:

myEndDate - End date of the ephemeris file.
Data Type: double
Privilege: Private
Default Value:

mySatelliteId - Identifies the satellite the ephemeris is based on.
Data Type: char array
Privilege: Private
Default Value:

mySecondsOfDayForEphemerisEnd - The seconds of day count.
Data Type: double
Privilege: Private
Default Value:

mySecondsOfDayForEphemerisStart - The seconds of day count.
Data Type: double
Privilege: Private
Default Value:

mySpaceCraftDataModeIndicator - The spacecraft data mode indicator is a mission dependent designation of the kind of data. The meaning of the data mode indicator for FDF products will be standardized.
Data Type: structure
Privilege: Private
Default Value:

mySpaceCraftInfo - Information about the spacecraft.
Data Type: structure
Privilege: Private
Default Value:

myStartDate - Start date of the ephemeris file.
Data Type: double
Privilege: Private
Default Value:

myTapeId - The tape identifier is always "standard" and is stored as the characters STANDARD.
Data Type: char array
Privilege: Private
Default Value:

myTimeSystemIndicator - The time system (atomic time, universal time coordinated (UTC)) used in the ephemeris file.
Data Type: char array
Privilege: Private
Default Value:

Operations:

ExtractAdditionalMetadata - Additional metadata are extracted in addition to metadata extraction at ingest to support certain services.
Arguments:
Return Type: Void
Privilege: Private

PrepareAdditionalMetadata - The Preprocessing "Prepare" operation prepares metadata required by the SDP Toolkit. Metadata that are not explicitly available may be derived from other lower level metadata (e.g., orbit number of L0 data files staged, etc.).
Arguments:
Return Type: Void
Privilege: Private

Reformat - The SDP Toolkit ephemeris tools require data to be in an uniform format independent of the source (FDF or spacecraft). The Preprocessing reformat functions perform the needed operations to convert data to a format acceptable to the SDP Toolkit. The data can also be converted to HDF-EOS. The most efficient format to handle ephemeris data is TBD.
Arguments:

Return Type: Void
Privilege: Private

Associations:

The DpPpFdfTrmmDefinitiveOrbitData class has associations with the following classes:
DpPpFdfData (Aggregation)
Class: DpPpFdfProcessingSet

4.4.57 DpPpLevelZeroData Class

Parent Class: DpPpPreprocessingData
Private
Persistent Class:
Purpose and Description:
The L0 products can be generalized based on the institutional source i.e., from TRMM spacecraft via SDPF and EOS-AM spacecraft via EDOS.

Attributes:

mySpaceCraftInfo - Name and information about the spacecraft.
Data Type: structure
Privilege: Private
Default Value:

Operations:

PrepareAdditionalMetadata - The Preprocessing "Prepare" operation prepares metadata required by the SDP Toolkit. Metadata that are not explicitly available may be derived from other lower level metadata (e.g. orbit number of L0 data files staged, etc.).
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpLevelZeroData class has associations with the following classes:
Class: DpPpSdpfLevelZeroProductionData

4.4.58 DpPpPreProcessing Class

The DpPrPreProcessing class provides an abstract view of the PreProcessing CSC and its respective role.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DpPpPreProcessing class has associations with the following classes:
Class: DpPrPgeExecutionManagement PerformsPre-ProcessingonStagedDataasaPGE

4.4.59 DpPpPreprocessingData Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This is a superclass that is a generalization of all ephemeris, L0 and external ancillary data.

Attributes:

myProductId - Name of the product for identification.
Data Type: char
Privilege: Private

Default Value:

myProject - This identifies the name of the project.
Data Type: char
Privilege: Private
Default Value:

mySourceId - The source ID of the file.
Data Type: char
Privilege: Private
Default Value:

Operations:

ExtractAdditionalMetadata - Additional metadata are extracted in addition to metadata extraction at ingest to support certain services.
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpPreprocessingData class has associations with the following classes:
None

4.4.60 DpPpQaParameters Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents the set of data quality processing parameters.

Attributes:

boxcarWindowSize - The number of ephemeris point to be included in the boxcar averaging window used to detect spikes in the ephemeris data.
Data Type: int
Privilege: Private

Default Value:

gapThreshold - The maximum duration of time over which the lack of ephemeris data can be tolerated.
Data Type: double
Privilege: Private
Default Value:

spikeThreshold - The maximum tolerable deviation of an ephemeris point from the normal trend.
Data Type: float
Privilege: Private
Default Value:

Operations:

None

Associations:

The DpPpQaParameters class has associations with the following classes:
None

4.4.61 DpPpQacList Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents the QAC list found in the Level Zero housekeeping dataset.

Attributes:

qacTable - This table contains the QAC flags referenced by sequential record number as ordered in the Level Zero Housekeeping dataset.
Data Type: int array
Privilege: Private
Default Value:

Operations:

DpPpQacList -
Arguments: fileIds:List<int>
Return Type: Void
Privilege: Private

getQacFlag - This lookups the QAC flag in the qacTable given the attitude packet record number.
Arguments: recordNumber:int
Return Type: Void
Privilege: Private

Associations:

The DpPpQacList class has associations with the following classes:
Class: DpPpAttitudeProcessingSet

4.4.62 DpPpSdpfLevelZeroDatasetFile Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
The SDPF Data Set Files are elements of a data product transferred in file format to the consumer. They are based on 24-hour data sets (6 hours in the case of Precipitation Radar APID) containing data generated by the TRMM spacecraft. A Data Set File consists of a Data Set File Header, a unique data set, and quality and accounting information for errored source data units. The 24-hour data are sorted and merged. The data packets received on virtual channels VC0, VC1 and VC11 (all housekeeping APIDs) will be contained within a single file, ordered by time only with redundant data removed, followed by a Quality and Accounting Capsule (QAC) list. Each QAC will contain information for a corresponding packet in the data set that was in error. Data packets received on all other VCs will be sorted by APID with one APID per file. The packets are sorted into forward source sequence count order, with redundant data removed, followed by the QAC list. Again, each QAC will contain information for a corresponding packet in the data set that was in error. There will be a Missing Data Units List (MDUL) at the end of each file. One Detached SFDU Header may reference multiple data set files.

Attributes:

myBeginningDateTime - This field indicates the data start time for this file.
Data Type: double
Privilege: Private
Default Value:

myDataType - This parameter identifies the data type of the data file (e.g., LZ means Level Zero, OR means orbit, AT means attitude, etc.).
Data Type: char
Privilege: Private
Default Value:

myDataVersion - Indicates the data version.
Data Type: char
Privilege: Private
Default Value:

myDescriptor - This parameter identifies the name of the instrument or sensor that collected the data, or further identified the type of data (e.g., SCR means spacecraft housekeeping, etc.).
Data Type: char
Privilege: Private
Default Value:

myDiscipline - Indicates the name of the discipline (e.g., Space Physics, etc.).
Data Type: char
Privilege: Private
Default Value:

myEndObjectFileGroup - This statement terminates the aggregation unit describing the attributes of a group of data files within the product.
Data Type: char
Privilege: Private
Default Value:

myEndObjectFileSpec - This statement terminates the aggregation unit describing an individual file within a data product.
Data Type: char
Privilege: Private
Default Value:

myEndingDateTime - This field indicates the data stop time for the file.
Data Type: double

Privilege: Private
Default Value:

myFileId - System file name for the specific data file described within the File_spec object.
Data Type: int
Privilege: Private
Default Value:

myGenerationDate - The time indicates the date and time of the generation of the data by the source system.
Data Type: double
Privilege: Private
Default Value:

myMission - This parameter indicates the mission or investigation which includes the sensors producing the data.
Data Type: char
Privilege: Private
Default Value:

myMissionParameters - Contains mission specific parameters.
Data Type: structure
Privilege: Private
Default Value:

myObjectFileGroup - Opens an aggregation of file group parameters; the attributes which characterize a set of data files within the product data.
Data Type: char
Privilege: Private
Default Value:

myObjectFileSpec - This statement opens an aggregation of file specific parameters: the attributes which characterize a particular data file within a file group. Each file group may contain multiple file specs, one for each specific data file.
Data Type: char
Privilege: Private
Default Value:

myProductInstance - Serial number of this instance of the SDPF to uniquely identify the data product.
Data Type: int
Privilege: Private
Default Value:

myProductName - Name of the SDPF product which defines the collection of files comprising the product.
Data Type: char
Privilege: Private
Default Value:

myProject - This name identifies the name of the project.
Data Type: char
Privilege: Private
Default Value:

myRecordSize - This parameter specifies the record size in bytes for this file.
Data Type: int
Privilege: Private
Default Value:

mySdpfSystem - ASCII string specifying the name of the SDPF mission serviced by the mission.
Data Type: char
Privilege: Private
Default Value:

mySequenceNumber - Sequence number created by SDPF to uniquely identify the data product.
Data Type: int
Privilege: Private
Default Value:

myTotalFileCount - This parameter indicates the total number of files for this product.
Data Type: int
Privilege: Private
Default Value:

Operations:

ExtractAdditionalMetadata - Additional metadata are extracted in addition to metadata extraction at ingest to support certain services.
Arguments:
Return Type: Void
Privilege: Private

PrepareAdditionalMetadata - The Preprocessing "Prepare" operation prepares metadata required by the SDP Toolkit. Metadata that are not explicitly available may be derived from

other lower level metadata (e.g. orbit number of L0 data files staged, number of L0 data files staged, pairing Standard Formatted Data Unit (SFDU) and Data Set File for TRMM processing, etc.).
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpSdpfLevelZeroDatasetFile class has associations with the following classes:
Class: DpPpSdpfLevelZeroSfduFile CorrespondsTo - Each SDPF generated SFDU file corresponds to a Data Set File.
DpPpSdpfLevelZeroProductionData (Aggregation)

4.4.63 DpPpSdpfLevelZeroProductionData Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
The L0 data from SDPF for CERES and LIS instruments for the TRMM spacecraft is a collection of Consultative Committee for Space Data Systems (CCSDS)-formatted telemetry packets. It will consist of L0 header and quality information parameters. The telemetry data packets contain instrument science data, spacecraft ancillary data and housekeeping or engineering data. The SDPF L0 Production Data Files correspond to separate Application Process Identifiers (APIDs). The SDPF L0 structure can be found in SDPF-TRMM Consumer ICD. This SDPF-generated production data is based on 24-hour data sets (6 hour data set for Precipitation Radar APID). The Production Data consists of an optional Standard Formatted Data Unit (SFDU) and Data Set File.

Attributes:

myBeginningDateTime - This field indicates the data start time for this file.
Data Type: double
Privilege: Private
Default Value:

myDataType - This parameter identifies the data type of the data file (e.g., LZ means L0, OR means orbit, AT means attitude, etc.).
Data Type: char

Privilege: Private
Default Value:

myDataVersion - Indicates the data version.
Data Type: char
Privilege: Private
Default Value:

myDescriptor - This parameter identifies the name of the instrument or sensor that collected the data, or further identifies the type of data (e.g., SCR means spacecraft housekeeping, etc.).
Data Type: char
Privilege: Private
Default Value:

myDiscipline - Indicates the name of the discipline (e.g., Space Physics, etc.)
Data Type: char
Privilege: Private
Default Value:

myDpcio - Data products content identifier object. To describe a "detached SFDU header" file, this DPCIO applies to a file group, and provides labels for the files within that file group.
Data Type: char
Privilege: Private
Default Value:

myEndObjectDpcio - This statement terminates the aggregation unit describing the "detached SFDU header."
Data Type: char
Privilege: Private
Default Value:

myEndObjectFileGroup - This statement terminates the aggregation unit describing the attributes of a group of data files within the product.
Data Type: char
Privilege: Private
Default Value:

myEndObjectFileSpec - This statement terminates the aggregation unit describing an individual file within a data product.
Data Type: char
Privilege: Private
Default Value:

myEndingDateTime - This field indicates the data stop time for this file.
Data Type: double
Privilege: Private
Default Value:

myFileId - System file name for the specific data file described within the File_spec object.
Data Type: char
Privilege: Private
Default Value:

myFileIdDpcio - This parameter when used with the DP_CIO indicates the system file name for the detached SFDU headers.
Data Type: char
Privilege: Private
Default Value:

myFileSize - This parameter when used with DP_CIO indicates the length in bytes of the DP_CIO.
Data Type: int
Privilege: Private
Default Value:

myGenerationDate - This time indicates the date and time of the generation of the data by the source system.
Data Type: double
Privilege: Private
Default Value:

myMission - This parameter indicates the mission or investigation which includes the sensors producing the data.
Data Type: char
Privilege: Private
Default Value:

myMissionParameters - Contains mission specific parameters.
Data Type: structure
Privilege: Private
Default Value:

myObjectFileGroup - Opens an aggregation of file group parameters; the attributes which characterize a set of data files within the product data.
Data Type: char
Privilege: Private
Default Value:

myObjectFileSpec - This statement opens an aggregation of file specific parameters: the attributes which characterize a particular data file within a file group. Each file group may contain multiple file specs, one for each specific data file.
Data Type: char
Privilege: Private
Default Value:

myProductInstance - Serial number of this instance of the SDPF to uniquely identify the data product.
Data Type: int
Privilege: Private
Default Value:

myProductName - Name of the SDPF product which defines the collection of files comprising the product.
Data Type: char
Privilege: Private
Default Value:

myProject - This name identifies the name of the project.
Data Type: char
Privilege: Private
Default Value:

myRecordSize - This parameter specifies the record size in bytes for this file.
Data Type: int
Privilege: Private
Default Value:

mySdpfSystem - ASCII string specifying the name of the SDPF mission service by the mission.
Data Type: char
Privilege: Private
Default Value:

mySequenceNumber - Sequence number created by SDPF to uniquely identify the data product.
Data Type: int
Privilege: Private
Default Value:

myTotalFileCount - This parameter indicates the total number of files for this product.
Data Type: int
Privilege: Private
Default Value:

Operations:

None

Associations:

The DpPpSdpfLevelZeroProductionData class has associations with the following classes:
Class: DpPpLevelZeroData

4.4.64 DpPpSdpfLevelZeroSfduFile Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
The SFDU (optional) consists of standard labels that uniquely identify and link a Data Set File to its description. The SFDU is referred to as Detached SFDU Header. There is one SFDU Header for each SDPF L0 product. The Detached SFDU Header consists of an SFDU Exchange Data Unit (EDU) Label, a Contents Identifier Object (CIO), and a Reference Identifier Object. One Detached SFDU Header may represent multiple Data Set Files.

Attributes:

myBeginningDateTime - This field indicates the data start time for this file.
Data Type: double
Privilege: Private
Default Value:

myDataType - This parameter identifies the data type of the data file (e.g., LZ means Level Zero, OR means orbit, AT means attitude, etc.).
Data Type: char
Privilege: Private
Default Value:

myDataVersion - Indicates the data version.
Data Type: char
Privilege: Private
Default Value:

myDescriptor - This parameter identifies the name of the instrument or sensor that collected the data, or further identifies the type of data (e.g., SCR means spacecraft housekeeping, etc.).
Data Type: char
Privilege: Private
Default Value:

myDiscipline - Indicates the name of the discipline (e.g., Space Physics, etc.).
Data Type: char
Privilege: Private
Default Value:

myDpcio - Data products content identifier object. To describe a "detached SFDU header" file, the DPCIO applies to a file group, and provides labels for the files within that file group.
Data Type: char
Privilege: Private
Default Value:

myEndObjectDpcio - This statement terminates the aggregation unit describing the detached SFDU header.
Data Type: char
Privilege: Private
Default Value:

myEndObjectFileGroup - This statement terminates the aggregation unit describing the attributes of a group of data files within the product.
Data Type: char
Privilege: Private
Default Value:

myEndObjectFileSpec - This statement terminates the aggregation unit describing an individual file within a data product.
Data Type: char
Privilege: Private
Default Value:

myEndingDateTime - This field indicates the data stop time for this file.
Data Type: double
Privilege: Private
Default Value:

myFileId - System file name for the specific data file described within the File_spec object.
Data Type: char
Privilege: Private
Default Value:

myFileIdDpcio - This parameter when used with the DP_CIO indicates the system file name for the detached SFDU headers.
Data Type: char
Privilege: Private
Default Value:

myFileSize - This parameter when used with DP_CIO indicates the length in bytes of the DP_CIO.
Data Type: int
Privilege: Private
Default Value:

myGenerationDate - This time indicates the date and time of the generation of the data by the source system.
Data Type: double
Privilege: Private
Default Value:

myMission - This parameter indicates the mission or investigation which includes the sensors producing the data.
Data Type: char
Privilege: Private
Default Value:

myMissionParameters - Contains mission specific parameters.
Data Type: structure
Privilege: Private
Default Value:

myObjectFileGroup - Opens an aggregation of file group parameters; the attributes which characterize a set of data files within the product data.
Data Type: char
Privilege: Private
Default Value:

myObjectFileSpec - This statement opens an aggregation of file specific parameters: the attributes which characterize a particular data file within a file group. Each file group may contain multiple file specs, one for each specific data file.
Data Type: char
Privilege: Private
Default Value:

myProductInstance - Serial number of this instance of the SDPF to uniquely identify the data product.
Data Type: int

Privilege: Private
Default Value:

myProductName - Name of the SDPF product which defines the collection of files comprising the product.
Data Type: char
Privilege: Private
Default Value:

myProject - This name identifies the name of the project.
Data Type: char
Privilege: Private
Default Value:

myRecordSize - This parameter specifies the record size in bytes for this file.
Data Type: int
Privilege: Private
Default Value:

mySdpfSystem - ASCII string specifying the name of the SDPF mission service by the mission.
Data Type: char
Privilege: Private
Default Value:

mySequenceNumber - Sequence number created by SDPF to uniquely identify the data product.
Data Type: int
Privilege: Private
Default Value:

myTotalFileCount - This parameter indicates the total number of files for this product.
Data Type: int
Privilege: Private
Default Value:

Operations:

ExtractAdditionalMetadata - Additional metadata are extracted in addition to metadata extraction at ingest to support certain services.
Arguments:
Return Type: Void
Privilege: Private

PrepareAdditionalMetadata - The Preprocessing "Prepare" operation prepares metadata required by the SDP Toolkit. Metadata that are not explicitly available may be derived from other lower level metadata (e.g. orbit number of L0 data files staged, number of L0 data files staged, pairing Standard Formatted Data Unit (SFDU) and Data Set File for TRMM processing, etc.).
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpSdpfLevelZeroSfduFile class has associations with the following classes:
Class: DpPpSdpfLevelZeroDatasetFile CorrespondsTo - Each SDPF generated SFDU file corresponds to a Data Set File.
DpPpSdpfLevelZeroProductionData (Aggregation)

4.4.65 DpPpTrmmOnBoardAttitudeData Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents attitude data within the TRMM spacecraft ancillary data contained in the SDPF-generated L0 data set.

Attributes:

myBeginningDateTime - See DpPpTrmmScAncillaryData class for description of all attributes.
Data Type: double
Privilege: Private
Default Value:

myDataType
Data Type: char
Privilege: Private
Default Value:

myDescriptor
Data Type: char

Privilege: Private
Default Value:

myDiscipline
Data Type: char
Privilege: Private
Default Value:

myEndingDateTime
Data Type: double
Privilege: Private
Default Value:

myFieldId -
Data Type: char
Privilege: Private
Default Value:

myFileId -
Data Type: char
Privilege: Private
Default Value:

myFileSize
Data Type: int
Privilege: Private
Default Value:

myGenerationDate
Data Type: double
Privilege: Private
Default Value:

myInstrumentName -
Data Type: char
Privilege: Private
Default Value:

myMission
Data Type: char
Privilege: Private
Default Value:

myMissionParameters
Data Type: structure

Privilege: Private
Default Value:

myProductInstance
Data Type: char
Privilege: Private
Default Value:

myProductName
Data Type: char
Privilege: Private
Default Value:

myProject
Data Type: char
Privilege: Private
Default Value:

myRecordSize
Data Type: int
Privilege: Private
Default Value:

mySequenceNumber
Data Type: int
Privilege: Private
Default Value:

mySpaceCraftInfo
Data Type: structure
Privilege: Private
Default Value:

Operations:

ExtractAdditionalMetadata - Additional metadata are extracted in addition to metadata extraction at ingest to support certain services.
Arguments:
Return Type: Void
Privilege: Private

PrepareAdditionalMetadata - The Preprocessing "Prepare" operation prepares metadata for the SDP Toolkit. Metadata that are not explicitly available may be derived from other

lower level metadata (e.g. orbit number of O/A data files staged, etc.).
Arguments:
Return Type: Void
Privilege: Private

QaCheck - The onboard orbit data are quality checked based on FDF provided specifications. Notifications to Ingest CI is made to get repaired orbit data if onboard orbit data do not satisfy FDF specifications.
Arguments:
Return Type: Void
Privilege: Private

Associations:

The DpPpTrmmOnBoardAttitudeData class has associations with the following classes:
DpPpTrmmScOaData (Aggregation)
Class: DpPpAttitudeProcessingSet

4.4.66 DpPpTrmmScAncillaryData Class

Parent Class: Not Applicable
Private
Persistent Class:
Purpose and Description:
This class represents data within the TRMM spacecraft ancillary packet contained in the SDPF generated L0 data.

Attributes:

myBeginningDateTime - This field indicates the data start time for this file.
Data Type: double
Privilege: Private
Default Value:

myDataType - This parameter identifies the data type of the data file (e.g., LZ means L0, OR means orbit, AT means attitude, etc.)
Data Type: char
Privilege: Private
Default Value:

myDescriptor - This parameter identifies the name of the instrument or sensor that collected the data, or further identifies the type of data (e.g., SCR means Spacecraft Housekeeping, etc.)
Data Type: char
Privilege: Private
Default Value:

myDiscipline - Indicates the name of the discipline (e.g., Space Physics, etc.)
Data Type: char
Privilege: Private
Default Value:

myEndingDateTime - This field indicates the data end time for this file.
Data Type: double
Privilege: Private
Default Value:

myFieldId - This parameter when used with DP_CIO indicates the system file name for the detached SFDU headers.
Data Type: char
Privilege: Private
Default Value:

myFileId - System file name for the specific data file described within the file specification object.
Data Type: char
Privilege: Private
Default Value:

myFileSize - This parameter when used with DP_CIO indicates the length in bytes of the DP_CIO.
Data Type: int
Privilege: Private
Default Value:

myGenerationDate - This time indicates the date and time of the generation of the data by the source system.
Data Type: double
Privilege: Private
Default Value:

myMission - This parameter indicates the mission or investigation which includes the sensors producing the data.
Data Type: char
Privilege: Private

Default Value:

myMissionParameters - Contains mission specific parameters.
Data Type: structure
Privilege: Private
Default Value:

myProductInstance - Serial number of this instance of the SDPF product.
Data Type: char
Privilege: Private
Default Value:

myProductName - Name of the SDPF product which defines the collection of files comprising the product.
Data Type: char
Privilege: Private
Default Value:

myProject - This name identifies the name of the project.
Data Type: char
Privilege: Private
Default Value:

myRecordSize - This parameter specifies the record size in bytes for this file.
Data Type: int
Privilege: Private
Default Value:

mySequenceNumber - Sequence number created by SDPF to uniquely identify the data product.
Data Type: int
Privilege: Private
Default Value:

Operations:

None

Associations:

The DpPpTrmmScAncillaryData class has associations with the following classes:
DpPpSdpfLevelZeroDatasetFile (Aggregation)

4.4.67 DpPpTrmmScOaData Class

Parent Class: DpPpEphemerisData
Private
Persistent Class:
Purpose and Description:
This class represents orbit/attitude data within the TRMM spacecraft ancillary packet contained in the SDPF generated Level zero data.

Attributes:

myBeginningDateTime - This filed indicates the data start time for this file.
Data Type: double
Privilege: Private
Default Value:

myDataType - This parameter identifies the data type of the data file (e.g., LZ means L0, OR means orbit, AT means attitude, etc.).
Data Type: char
Privilege: Private
Default Value:

myDescriptor - This parameter identifies the name of the instrument or sensor that collected the data, or further identifies the type of data (e.g., SCR Spacecraft, Housekeeping, etc.).
Data Type: char
Privilege: Private
Default Value:

myDiscipline - Indicates the name of the discipline (e.g., Space Physics, etc.).
Data Type: char
Privilege: Private
Default Value:

myEndingDateTime - This field indicates the data end time for this file.
Data Type: double
Privilege: Private
Default Value:

myFieldId - This parameter when used with DP_CIO indicates the system file name for the detached SFDU headers.
Data Type: char
Privilege: Private
Default Value:

myFileId - System file name for the specific data file described within the file specification object.
Data Type: char
Privilege: Private
Default Value:

myFileSize - This parameter when used with DP_CIO indicates the length in bytes of the DP_CIO.
Data Type: int
Privilege: Private
Default Value:

myGenerationDate - This time indicates the date and time of the generation of the data by the source system.
Data Type: double
Privilege: Private
Default Value:

myMission - This parameter indicates the mission or investigation which includes the sensors producing the data.
Data Type: char
Privilege: Private
Default Value:

myMissionParameters - Contains mission specific parameters.
Data Type: structure
Privilege: Private
Default Value:

myProductInstance - Serial number of this instance of the SDPF product.
Data Type: char
Privilege: Private
Default Value:

myProductName - Name of the SDPF product which defines the collection of files comprising the product.
Data Type: char
Privilege: Private
Default Value:

myProject - This name identifies the name of the project.
Data Type: char
Privilege: Private
Default Value:

myRecordSize - This parameter specifies the record size in bytes for this file.
Data Type: int
Privilege: Private
Default Value:

mySequenceNumber - Sequence number created by SDPF to uniquely identify the data product.
Data Type: int
Privilege: Private
Default Value:

Operations:

All Operations inherited from parent class

Associations:

The DpPpTrmmScOaData class has associations with the following classes:
DpPpTrmmScAncillaryData (Aggregation)

4.5 CSCI Dynamic Model

As part of the PDPS detailed design process, the group of existing scenarios which were developed for the Preliminary Design Specification have been updated. As a result of the changes in the boundaries of Planning and Processing and the selection of AutoSys and AutoXpert products, the scenarios have undergone a great deal of modification. The basic scenario for initiation of a job as performed by AutoSys is presented in Section 4.1.4 and accompanying Figures 4.1-1 and 4.1-2. This information should be referenced as these other scenarios are reviewed. Before the scenarios, an introduction section has been provided which provide a list of assumptions used in developing the scenarios. These assumptions explain the interfaces and the given state of the ECS system and subsystems.

The Processing CSCI Dynamic Model is represented by a group of scenarios and event traces which have been developed to further refine the role, responsibilities and activities of the Processing CSCI during SDPS operations. The scenarios provide an abstract section to describe the system context of the processing, and a detailed scenario description of the required Processing CSCI activities. Event Traces have been developed for each scenario to show the interactions between the objects involved in the scenario. There may be more than one event trace for each scenario depending on the complexity of processing represented in the scenario. Because of the COTS-intensive aspects of the Processing CSCI design, the scenarios provide descriptions of events which occur in the custom code which is being developed to support the COTS' products. In the scenario, a brief description of the COTS activities is provided, but within the event traces, the COTS is treated as a black box. Each scenario provides a listing of the classes from the object model referenced in the scenario. The scenarios have been grouped around significant Processing CSCI functional areas.

These groupings are as follows:

a. Job Management Scenarios—These scenarios are presented to explain certain concepts to understand the integrated view between AutoSys and the Processing CSCI custom components. A scenario is provided which describes how information provided in a Data Processing Request is used to create the jobs required to execute a PGE. Also provided are scenarios which described the Processing CSCI activities involved with the input of a daily schedule of Data Processing Requests from Planning, providing status information to Planning, activating jobs for execution, updating job information, and canceling of jobs.

b. Data Management Scenarios—These scenarios describe the Processing CSCI activities which occur to support the following activities:

   1. Staging and destaging of data from and to the Data Server.
   2. Retaining of data on Science Production Hardware to support further production
   3. Deletion of data not required to support further production.
   4. Movement of data from one production resource to another to support further production.
   5. Support for pre-processing of data for production.

c. PGE Execution Management Scenarios—These scenarios describe the activities which occur to initiate and manage the execution of a PGE.

d. Resource Management Scenarios - These scenarios describe activities related to the management of Processing CSCI hardware resources.

e. Quality Assurance Scenarios - These scenarios present high-level view of the activities which will support the DAAC manual quality assurance activities.

4.5.1 Scenario Assumptions

The following information details ECS system assumptions which should be followed while reviewing these scenarios. These assumptions define information on Processing CSCI interfaces as well as subsystem and system state information. Unless specified otherwise, this information remains consistent for each scenario. For the event traces, the Job Scheduling COTS is represented as a class. This is an abstract class and is used to represent the stimuli which result in Processing CSCI activities.

4.5.1.1 Interfaces

Table 4.5-1 shows the service of processing CSCI interfaces with other subsystems.

Table 4.5-1. Processing CSCI Interfaces With Other Subsystems

Subsystem	Service
CSS	Supplies the communication mechanisms used between ECS Applications. These mechanisms are based on OODCE services. Processing CSCI uses these mechanisms to communicate with MSS, Data Server, and Ingest.
MSS	1) Provides services to startup and shutdown ECS applications. MSS would initiate activities to startup and shutdown the PDPS Database, AutoSys' Database, AutoSys' Event Processor, and other required components needed at startup. Also, in the event of a shutdown, MSS would shutdown the components in the required order. 2) Provides services to log system events, i.e. Fault and exception handling information, in order to notify all impacted parties.
Data Server	Provides services to stage (Acquire) and destage (Insert) data from Processing hardware. Also, the Q/A Monitor Operations position requires subscription services of the Data Server.
Ingest	Same as Data Server. Ingest is a specialized Data Server
Planning	Provides Data Processing Request information to Processing. The interface mechanisms occur through AutoSys provided command-line interfaces and APIs and through the PDPS Database.

4.5.1.2 System and Subsystem States

During these scenarios, all ECS application components are considered to be operating in a steady state fashion, unless specified otherwise. Some of the scenarios do deal with failure situations and the results of these failures.

4.5.2 Job Management Scenarios

These scenarios describe the activities associated with managing the AutoSys job schedule. The activities discussed include creation of the job schedule, cancellation of a job, modification of a job or providing of status of a job. Also, the scenarios explain the interface which exists between the Planning CSCI and the COTS (AutoSys) component of the Processing CSCI. Each of these activities is initiated by a Planning CSCI component. A brief overview of Planning CSCI activities follows.

The Production Planner (Operations Person) is responsible for the creation of candidate production plans. These plans provide predicted views of science data production based on certain criteria, such as resource availability schedules and different standard production priorities. Once the Production Planner decides on the best candidate plan, this candidate plan is activated. (See the Planning Subsystem Detailed Design Specification for more information on these Planning activities). According to current DAAC Operations' strategies, a section of this active plan, known as the daily job schedule, will be fed into the AutoSys Database at the beginning of the day. These jobs are the jobs which will be processed for a particular day. Since the Preliminary Design Specification, an important change has taken place in the interface between the Planning and Processing CSCIs. This change involves when a Data Processing Request is made visible to the Processing CSCI. At PDR, the approach amounted to not providing a Data Processing Request to the Processing CSCI until all the data dependencies were fulfilled for that Data Processing Request. Because of the selection of AutoSys and its capabilities to manage job dependencies, this approach has been changed to consist of all Data Processing Requests being fed into AutoSys at the beginning of the day. The Data Processing Requests which do not have all data dependencies fulfilled would be kept in a "HELD" state until the dependencies are fulfilled. Upon the meeting of all data dependencies, the Planning CSCI would release the job. To limit the reach of the AutoSys specific interfaces, a layer of software, sometimes referred to as Glue Software, has been developed to promote the integration of AutoSys with ECS custom software applications. This allows Planning and Processing CSCI component data structures and operation primitives to be associated with AutoSys specific primitives with limited impact to the overall implementation of these components. In the Processing CSCI, this Glue Software is represented by the classes which are part of the Job Management component. One of these classes, DpPrCotsManager, encapsulates the AutoSys specific command-line interfaces and APIs used to communicate with AutoSys.

4.5.2.1 Create Data Processing Request (DPR) or Ground Event Job

4.5.2.1.1 Abstract

Prior to the activation of a plan, a candidate plan is created and deemed acceptable to be used as the predicted science data production plan. The Production Planner (Operations Person) activates the Production Planning Workbench. In a previous session with the workbench, the Production Planner had created a candidate plan which will now be activated. By choosing the Activate Plan command, the Production Planner initiates the Activate Plan operation. All Data Processing Requests and Ground Events which reside in the part of the active plan for the chosen daily schedule time frame will be added to the AutoSys job schedule for further processing.

A Ground Event Job is the method used by Planning to make resources unavailable to support production. A Ground Event is input by the Resource Planner who uses Planning CSCI software to make a Resource Availability Schedule. This schedule is made available to the Planning CSCI and is used as an input to the Production Plan. To schedule a resource for maintenance, system test, or some other defined ground event, a ground event job is input into the AutoSys Job Schedule.

For a given Data Processing Request or Ground Event, an AutoSys Job Box, which contains a collection of related jobs required to successfully process the Data Processing Request/Ground Event, is created. These jobs will be added to the AutoSys job stream. Job definition and dependency information, such as priority, start times, required resources and the execution status of other jobs, will be used to determine when a job should begin processing.

The series of steps in the following scenario description are repeated for each Data Processing Request/Ground Event which is scheduled for processing on a particular day.

4.5.2.1.2 Stimulus

Production Planner initiates daily job schedule process. The operations CreateDPRjob and CreateGEvnt associated with the class DpPrScheduler are used to communicate with the AutoSys components. The DpPrScheduler and associated classes encapsulate the specific command-line interfaces and APIs used to communicate with AutoSys.

4.5.2.1.3 Desired Response

A series of jobs required to execute a PGE are created. These jobs perform the following activities:

a. Data Staging (DPR only)

b. Resource Allocation

c. Execution of the PGE (DPR only)

d. Data Destaging (DPR only)

f. Resource Deallocation

Log Information to record the event will be collected and stored for review at a later time. These jobs will be added to the AutoSys Database and will become part of the AutoSys job stream.

4.5.2.1.4 Participating Classes from the Object Model

a. PlDpr

b. DpPrScheduler

c. DpPrCotsManager

d. Cots

e. DpPrDataManager (DPR only)

f. PlPge (DPR only)

g. PlGroundEvent (Ground Event only)

4.5.2.1.5 Description

1) When the CreateDPRJob operation is initiated on the DpPrScheduler, the following steps will occur:

   a) The DpPrScheduler will retrieve information as needed from the PlDpr and associated PlPge to perform the following steps for each DPR:

      (1) Create a series of jobs in AutoSys' job schedule (part of AutoSys' database schema) which parallel the requested DPR processing; these jobs are organized into a "job box" which represents the DPR at a higher level.

      (2) Create dependency information in AutoSys

      (3) Create other pre and post processing events or jobs which will initiate additional Processing software, such as Data Management, PGE Execution Management, or Resource Management CSC operations.

   b) The jobs added are considered "on hold" until Planning, through the Job Scheduler class, performs a release job operation. Planning will immediately release the job (see Release Data Processing Request (DPR) Job scenario).

   c) END OF SCENARIO.

2) When the CreateGEvnt operation is initiated on the DpPrScheduler, the following steps will occur:

   a) The DpPrScheduler will retrieve information as needed from the PlGroundEvent to perform the following steps:

      (1) Create a series of jobs in AutoSys' job schedule (part of AutoSys' database schema) which parallel the requested Ground Event processing; these jobs

are organized into a "job box" which represents the Ground Event at a higher level.

b) The Job Box is scheduled to start based on the time information from the Ground Event.

c) END OF SCENARIO.

4.5.2.1.6 Event Traces

4.5.2.2 Release Data Processing Request (DPR) Job

4.5.2.2.1 Abstract

When a Data Processing Request is fed into AutoSys through the CreateDPRJob operation on the DpPrScheduler class, the resulting job box is set in the "HELD" state in the AutoSys Database. This results in the jobs not being processed, or managed, until the job box (referred to herein as the "DPR job") is released. The Planning CSCI will release a DPR job when all the data subscriptions for that DPR are fulfilled. Therefore, for all Data Processing Requests which have no outstanding data subscriptions, the DPR jobs will be initiated as "HELD" and then immediately released. Other DPR jobs which have outstanding data subscriptions will reside in the "HELD" state until Planning has been notified that the data is available from the Science Data Server.

When the Planning CSCI has determined that no outstanding data subscriptions exist for a DPR job, that DPR job is released by using the ReleaseDPRJob operation in the DpPrScheduler class. The result of this operation will be that the DPR job and the individual jobs within it become active in the AutoSys job stream, and will be processed accordingly. The Scheduling COTS product examines the released job to see if it has any dependencies on previous jobs; if so, and those dependencies have not been satisfied, the current DPR is prevented from running until they are. If those dependencies have been met, the jobs in the job box start executing. (See Figure 4.5-1 and Figure 4.5-2.)

The series of steps in the following scenario description are repeated for each Data Processing Request which is scheduled for processing for on a particular day.

4.5.2.2.2 Stimulus

Upon receipt of the last outstanding Subscription Notification for a Data Processing Request, the Planning CSCI will use the ReleaseDPRJob operation in the DpPrScheduler class. The DpPr-Scheduler and related classes encapsulate the AutoSys specific command-line interfaces and APIs to used to communicate with AutoSys.

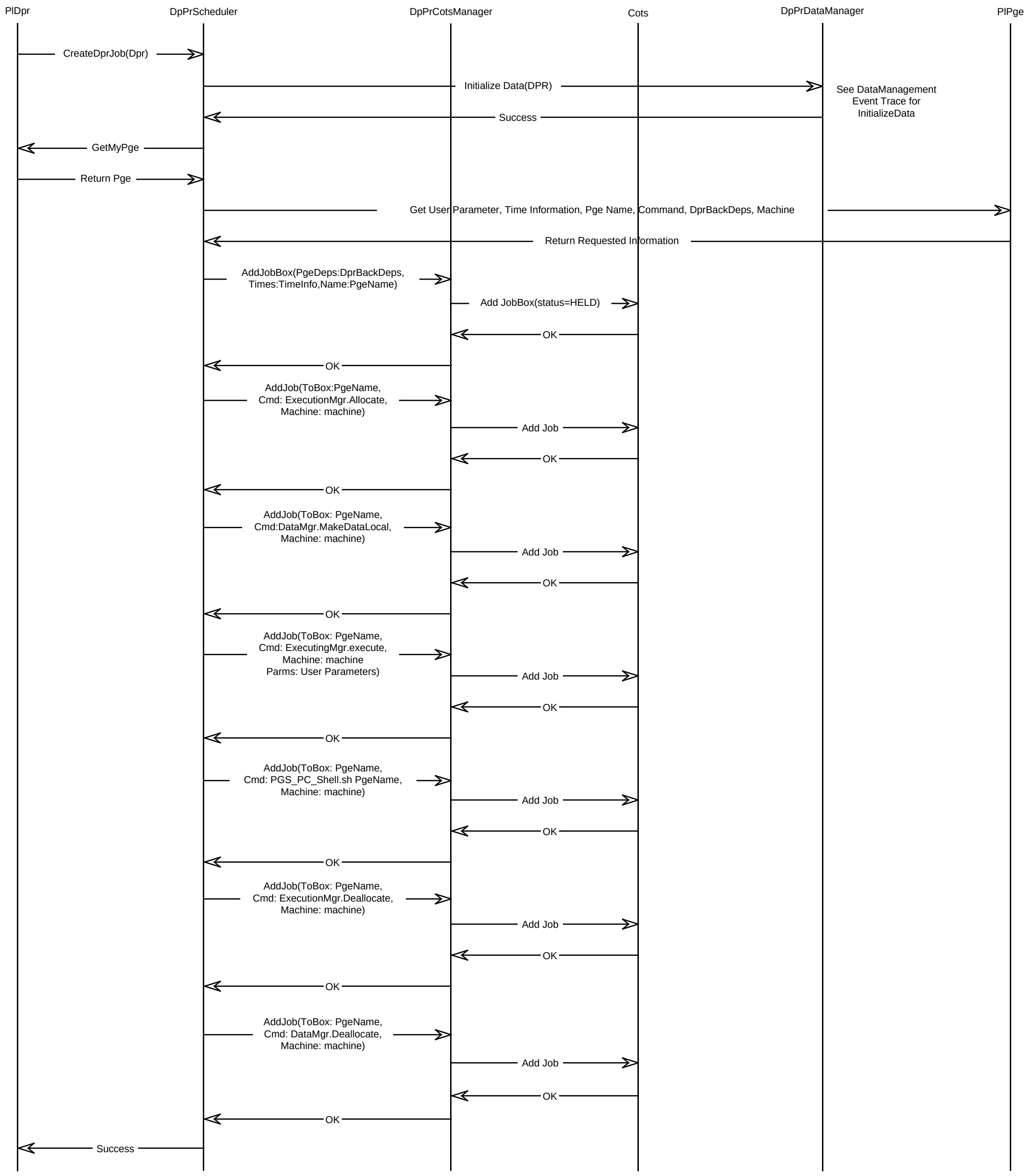

PlDpr
DpPrScheduler
DpPrCotsManager
Cots
DpPrDataManager
PlPge
CreateDprJob(Dpr)
Initialize Data(DPR)
See DataManagement Event Trace for InitializeData
Success
GetMyPge
Return Pge
Get User Parameter, Time Information, Pge Name, Command, DprBackDeps, Machine
Return Requested Information
AddJobBox(PgeDeps:DprBackDeps, Times:TimeInfo,Name:PgeName)
Add JobBox(status=HELD)
OK
OK
AddJob(ToBox:PgeName, Cmd: ExecutionMgr.Allocate, Machine: machine)
Add Job
OK
OK
AddJob(ToBox: PgeName, Cmd:DataMgr.MakeDataLocal, Machine: machine)
Add Job
OK
OK
AddJob(ToBox: PgeName, Cmd: ExecutingMgr.execute, Machine: machine Parms: User Parameters)
Add Job
OK
OK
AddJob(ToBox: PgeName, Cmd: PGS_PC_Shell.sh PgeName, Machine: machine)
Add Job
OK
OK
AddJob(ToBox: PgeName, Cmd: ExecutionMgr.Deallocate, Machine: machine)
Add Job
OK
OK
AddJob(ToBox: PgeName, Cmd: DataMgr.Deallocate, Machine: machine)
Add Job
OK
OK
Success

Figure 4.5-1. Create Data Processing Request Job Event Trace

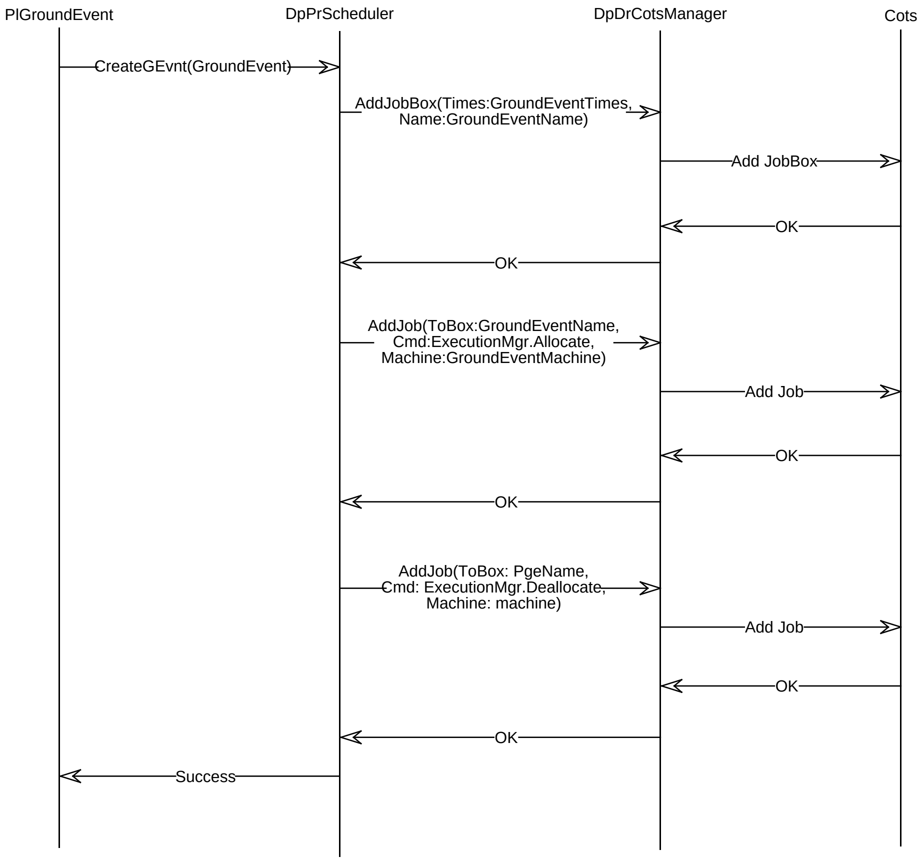

PlGroundEvent
DpPrScheduler
DpDrCotsManager
Cots
CreateGEvnt(GroundEvent)
AddJobBox(Times:GroundEventTimes, Name:GroundEventName)
Add JobBox
OK
OK
AddJob(ToBox:GroundEventName, Cmd:ExecutionMgr.Allocate, Machine:GroundEventMachine)
Add Job
OK
OK
AddJob(ToBox: PgeName, Cmd: ExecutionMgr.Deallocate, Machine: machine)
Add Job
OK
OK
Success

Figure 4.5-2. Create Ground Event Job Event Trace

4.5.2.2.3 Desired Response

The specified DPR job and the individual jobs within it are activated in the AutoSys job stream and processed accordingly. Log Information to record the event will be collected and stored for review at a later time.

4.5.2.2.4 Participating Classes from the Object Model

a. PlDpr

b. DpPrScheduler

c. DpPrCotsManager

d. Cots

e. DpPrDataManager

f. DpPrExecutionManager

g. DpPrExecutable

4.5.2.2.5 Description

1) When the ReleaseDPRJob operation is initiated on the DpPrScheduler, the following steps will occur:
   a) The DpPrScheduler will release the job in the AutoSys Database. Figure 4.5-3, Release Data Processing Request Event Trace, shows the events to needed to support this step.
2) If the DPR Job's dependencies on prior jobs have been fulfilled, the Job Box will begin to execute. Figure 4.5-4, Job Box Execution, shows the events to needed to support this step.
3) END OF SCENARIO.

4.5.2.2.6 Event Traces

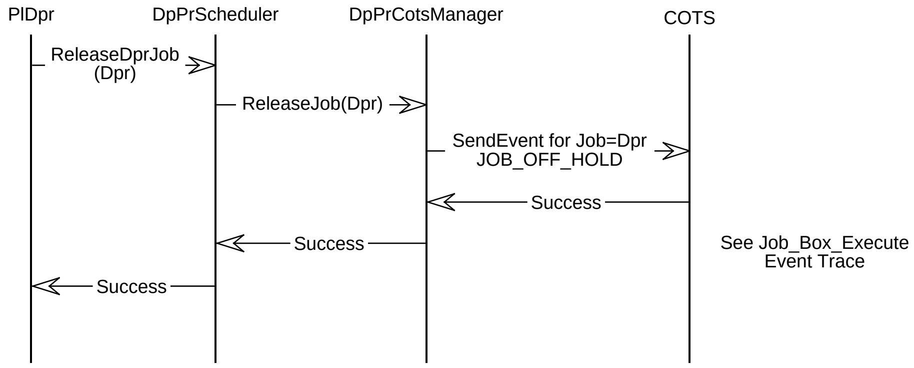


Figure 4.5-3. Release Data Processing Request Job Event Trace

4.5.2.3 Cancel Data Processing Request or Ground Event Job

4.5.2.3.1 Abstract

This scenario describes the processing required to cancel further processing of a Data Processing Request or Ground Event. The Planning CSCI will cancel a DPR job/Ground Event job when the job's associated Request/Event is no longer relevant. This could occur when the Production Plan has been modified or when the processing of a particular Production Request has been discontinued. The result of the cancellation is that the processing of the job will be halted. No further processing for this job will be performed unless the job is re-created and added back into the AutoSys job stream.

For Data Processing Request jobs, if the PGE associated with the Data Processing Request was executing at the time of cancellation, this execution will be terminated. If the PGE was not executing, but the data staging process was in progress, this process will also be terminated, and all resources reserved to process this PGE will be freed.

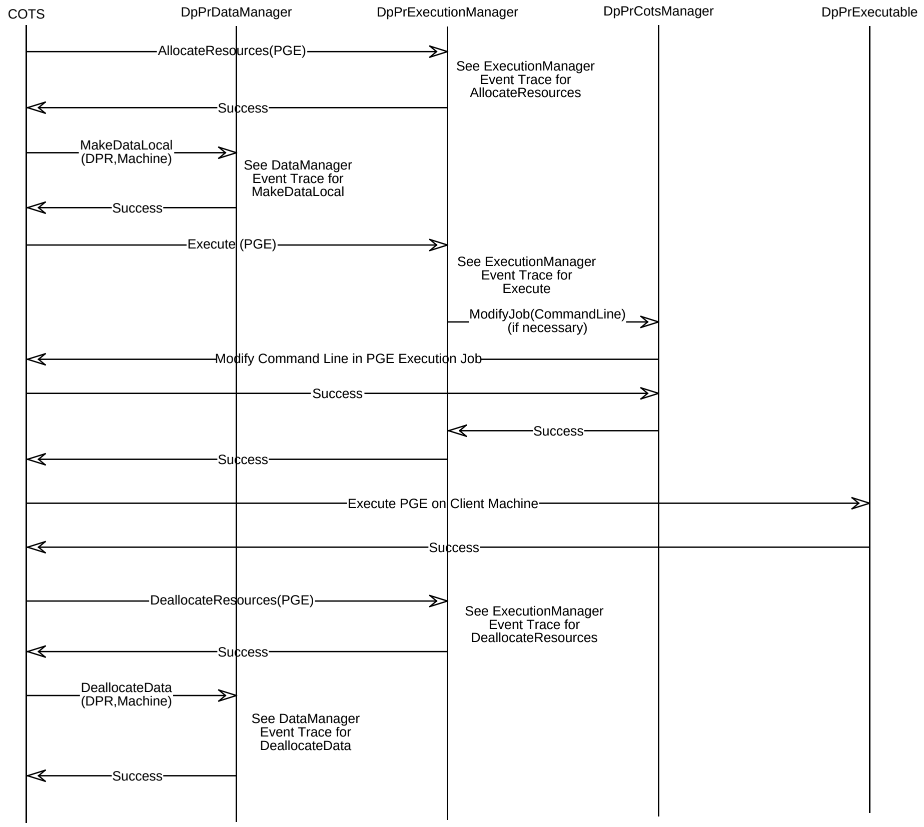

COTS
DpPrDataManager
DpPrExecutionManager
DpPrCotsManager
DpPrExecutable
AllocateResources(PGE)
See ExecutionManager Event Trace for AllocateResources
Success
MakeDataLocal (DPR,Machine)
See DataManager Event Trace for MakeDataLocal
Success
Execute (PGE)
See ExecutionManager Event Trace for Execute
ModifyJob(CommandLine) (if necessary)
Modify Command Line in PGE Execution Job
Success
Success
Success
Execute PGE on Client Machine
Success
DeallocateResources(PGE)
See ExecutionManager Event Trace for DeallocateResources
Success
DeallocateData (DPR,Machine)
See DataManager Event Trace for DeallocateData
Success

Figure 4.5-4. Job Box Execution Event Trace

A Ground Event Job is the method used by Planning to make resources unavailable to support production. A Ground Event is input by the Resource Planner who uses Planning CSCI software to make a Resource Availability Schedule. This schedule is made available to the Planning CSCI and is used as an input to the Production Plan. To schedule a resource for maintenance, system test, or some other defined ground event, a ground event job is input into the AutoSys Job Schedule.

4.5.2.3.2 Stimulus

The initiation of the CancelDPRJob or CancelGEvnt operation on the JobScheduler class.

4.5.2.3.3 Desired Response

The following actions will occur during Cancel Data Processing Request processing:

a. The execution of the PGE associated with the Data Processing Request will be terminated, if executing, or canceled, if awaiting execution.

b. The COTS operations display will be updated.

c. If allocation of resources had occurred to support the execution of the PGE or the implementation of the Ground Event, the allocated resources will be freed.

d. Log Information to record the event will be collected and stored for review at a later time

4.5.2.3.4 Participating Classes From the Object Model

a. PlDpr (DPR only)

b. DpPrScheduler

c. DpPrCotsManager

d. Cots

e. DpPrDataManager (DPR only)

f. DpPrExecutionManager

g. PlGroundEvent (Ground Event only)

4.5.2.3.5 Scenario Description

a) When the CancelDPRJob or CancelGEvnt operation is initiated on the DpPrScheduler, the following steps will occur:

1) The DpPrScheduler will cancel the job in the AutoSys Database.

2) If the PGE associated with Data Processing Request is executing, the PGE is terminated. No attempt is made to complete execution. By indicating cancellation, Planning has indicated that the processing should not be accomplished.

3) If resources (such as disk space or CPU time) have been reserved for the PGE or Ground Event, these will be deallocated so another job can use them.

b) END OF SCENARIO.

4.5.2.3.6 Event Traces

Figure 4.5-5, Cancel Data Processing Request Job Event Trace, show the steps required to cancel a Data Processing Request job. Figure 4.5-6, Cancel Ground Event Job Event Trace, show the steps required to cancel a Ground Event job. The Ground Event Job is used to take resources out of the active job schedule so that maintenance, system test, or special events can be supported.

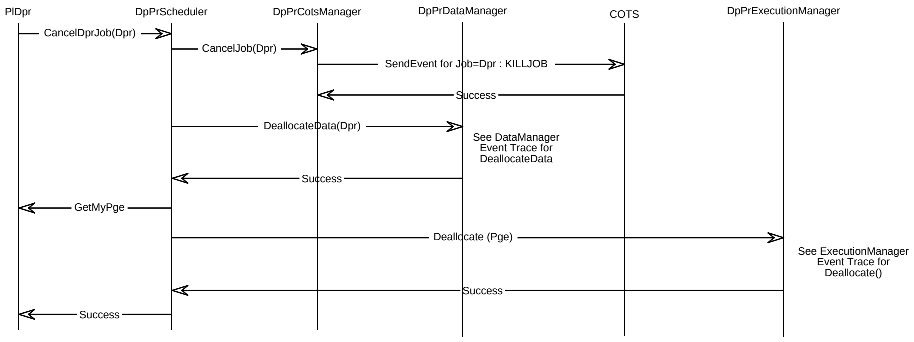


Figure 4.5-5. Cancel Data Processing Request Job Event Trace

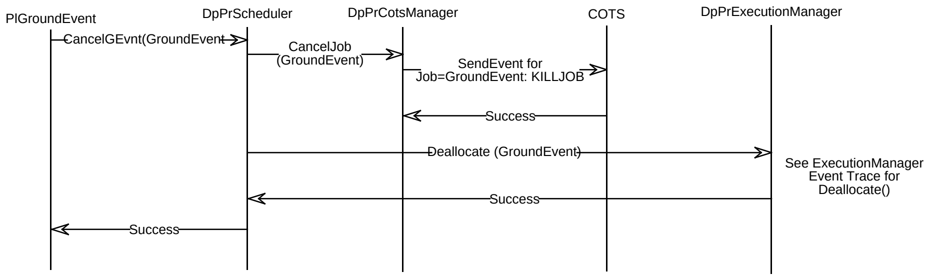


Figure 4.5-6. Cancel Ground Event Job Event Trace

4.5.2.4 Update Data Processing Request Job

4.5.2.4.1 Abstract

This scenario describes the processing required to update the AutoSys job definition associated with a Data Processing Request. This process is initiated through the UpdateDPRJob operation in the DpPrScheduler class. The job definition information available for modification includes priority information and time information, such as start times, predicted execution times and ending time restrictions. Other Data Processing Request information can only be modified through a Planning CSCI interface.

4.5.2.4.2 Stimulus

The Planning CSCI will use the UpdateDPRJob operation in the DpPrScheduler class. The DpPrScheduler and associated classes encapsulate the AutoSys specific command-line interfaces and APIs to used to communicate with AutoSys.

4.5.2.4.3 Desired Response

The job definition of the Data Processing Request which resides in the AutoSys Database will be modified. AutoSys will add a log entry to its database to record this event.

4.5.2.4.4 Participating Classes from the Object Model

a. PlDpr

b. DpPrScheduler

c. DpPrCotsManager

d. Cots

4.5.2.4.5 Description

a) When the UpdateDPRJob operation is initiated on the DpPrScheduler, the following steps will occur:

1) The DpPrScheduler will update the job definition associated with the Data Processing Request in the AutoSys Database.

b) END OF SCENARIO.

4.5.2.4.6 Event Trace

Figure 4.5-7 shows the update data processing request job event trace.

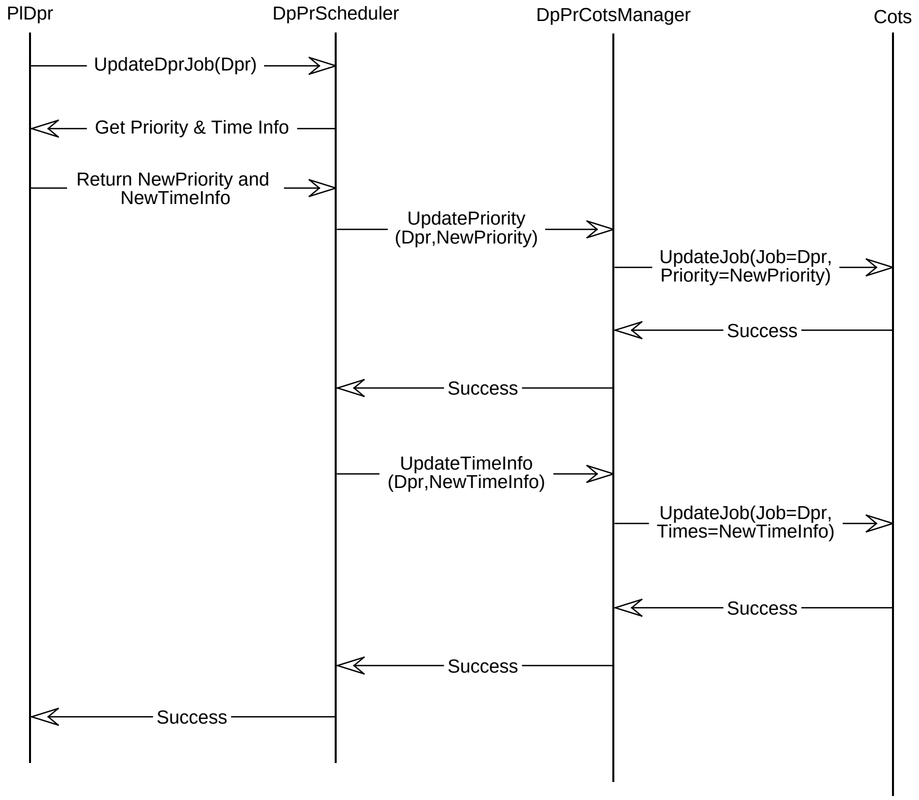

PlDpr
DpPrScheduler
DpPrCotsManager
Cots
UpdateDprJob(Dpr)
Get Priority & Time Info
Return NewPriority and NewTimeInfo
UpdatePriority (Dpr,NewPriority)
UpdateJob(Job=Dpr, Priority=NewPriority)
Success
Success
UpdateTimeInfo (Dpr,NewTimeInfo)
UpdateJob(Job=Dpr, Times=NewTimeInfo)
Success
Success
Success

Figure 4.5-7. Update Data Processing Request Job Event Trace

4.5.2.5 Get Data Processing Request Job Status

4.5.2.5.1 Abstract

This scenario describes the processing required to provide status information about the processing of a Data Processing Request to the Planning CSCI. The status of a Data Processing Request is provided during the AutoSys defined life of a Data Processing Request. This status information will provide the Planning CSCI a mechanism to judge the actual performance of the active plan compared to the predicted plan activities. Also, this will be used by Planning as a means of updating the Plan to determine the next group of activities to be provided to AutoSys.

4.5.2.5.2 Stimulus

The Planning CSCI will use the GetDPRJobStatus operation in the DpPrScheduler class. The DpPrScheduler and associated classes encapsulate the AutoSys specific command-line interfaces and APIs to used to communicate with AutoSys.

4.5.2.5.3 Desired Response

The status of the DPR job in the AutoSys job stream will be returned to Planning.

4.5.2.5.4 Participating Classes from the Object Model

a. PlDpr

b. DpPrScheduler

c. DpPrCotsManager

d. Cots

4.5.2.5.5 Description

a. When the GetDPRJobStatus operation is initiated on the DpPrScheduler, the following steps will occur:

1. Retrieve job status information from the AutoSys database.
2. Create Status Report.
3. Return Status Report to the Planning CSCI.

b. End of Scenario

4.5.2.5.6 Event Traces

Figure 4.5-8 shows the get data processing request job status event trace.

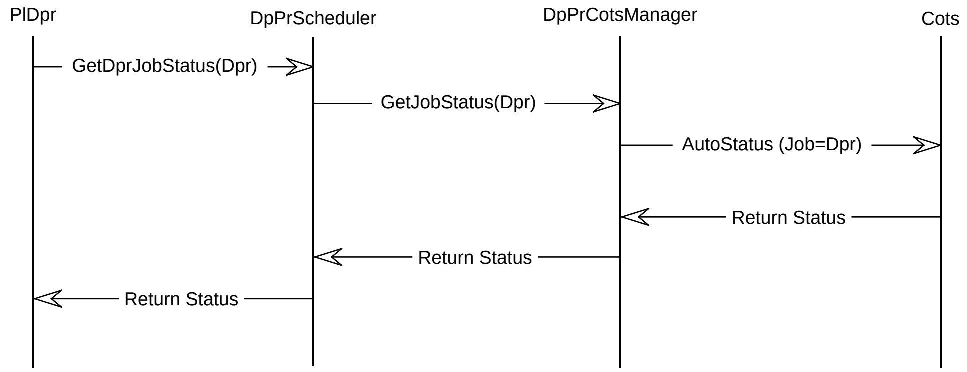


Figure 4.5-8. Get Data Processing Request Job Status Event Trace

4.5.3 Data Management Scenarios

These scenarios describe the activities associated with the Data Management component of the Processing CSCI. These scenarios describe the Processing CSCI activities which occur to support the following activities:

1. Staging and destaging of data from and to the Science Data Server CSCI.
2. Retaining of data on Science Production Hardware to support further production
3. Deletion of data not required to support further production.
4. Movement of data from one production resource to another to support further production.

These scenarios explain the interface which exists between the Processing CSCI and the Science Data Server CSCI. Data Staging and Destaging are services used to interface with the Science Data Server to coordinate the transfer of data. Data Staging defines the transfer of data from the Science Data Server CSCI to the Processing CSCI, and Data Destaging defines the transfer of data from the Processing CSCI to the Science Data Server CSCI. The protocol used to provide these services consists of the Processing CSCI requesting the transfer of data from the Science Data Server to Processing or from Processing to the Science Data Server. The Science Data Server is responsible for the movement of the data. Any data requiring transfer from the Science Data Server to Processing, or from Processing to the Science Data Server will be transferred using this approach. This data includes PGE scripts, algorithm executables, PGE status message files, PGE process control files, metadata, calibration data files, ancillary data products, or ECS Data Products. At this time, the Processing design is based on the assumption that all data staging occurs prior to the start of PGE execution. Also, PGE execution is not considered complete until destaging of the output data products occurs.

As a result of the initiation of data staging by Processing, the Science Data Server will initiate an FTP PUSH operation to copy the data to the science processing hardware. As a result of the initiation of data destaging by Processing, the Science Data Server will initiate an FTP PULL operation to copy the data from the science processing hardware to the Data Server subsystem hardware.

Please note that the Processing CSCI uses the Science Data Server CSCI provided services to stage data from the Ingest subsystem as well as the Data Server Subsystem. Also, the Processing CSCI interfaces with Data Servers at different DAAC locations to stage and destage data.

4.5.3.1 Data Initialization

4.5.3.1.1 Abstract

The Processing CSCI has the responsibility to manage data that is currently residing on the science processing hardware resources and to retain data that is required to support the execution of multiple PGEs. To manage this data, the Processing CSCI must retain knowledge of the current location of data. The location may be at the Data Server or on some local science Processing hardware resource. To retain knowledge of the current location of data and other characteristics that are used to define a data object, the Processing CSCI has established a persistent class referred to as the DpPrDataMap. The persistence of this class is retained through the use of the PDPS Database. When this class is required to support a Processing CSCI operation, this class is created and information is extracted from the PDPS Database. More information on the PDPS Database and the extraction of data is contained in the Planning Subsystem Detailed Design Specification. The information retained in this class is used during different Processing CSCI operations to determine if staging or destaging of a given data object is required.

The initialization of the class DpPrDataMap occurs when the CreateDPRJob operations of DpPrScheduler class is activated. Part of the CreateDPRJob operation will consist of creating the DpPrDataMap for a Data Processing Request.

4.5.3.1.3 Stimulus

When the Planning CSCI adds a Data Processing Request to AutoSys' daily job schedule, the DpPrScheduler will abstract data from the Data Processing Request which will be retained in the DpPrDataMap class. The persistence of this data will be managed through the PDPS Database.

4.5.3.1.4 Desired Response

The initialization of the DpPrDataMap class which contains information on the characteristics of data that a PGE associated with a Data Processing Request requires for execution.

4.5.3.1.5 Participating Classes From the Object Model

- DpPrScheduler
- DpPrDataManager
- DpPrDataMap
- PlDPR
- PlDataGranule
- GlUR

4.5.3.1.8 Scenario Description

a) When the CreateDPRJob operation is initiated on the DpPrScheduler, the following steps will occur:

   1) The DpPrScheduler will retrieve information as needed from the PDPS database to perform the following steps for each job:

      (a) Create a DpPrDataMap used to retain information on the input data required to support the execution of the PGE associated with the Data Processing Request.

b) END OF SCENARIO.

4.5.3.1.9 Event Trace

Figure 4.5-9 shows the DpPrDataMap Initialization.

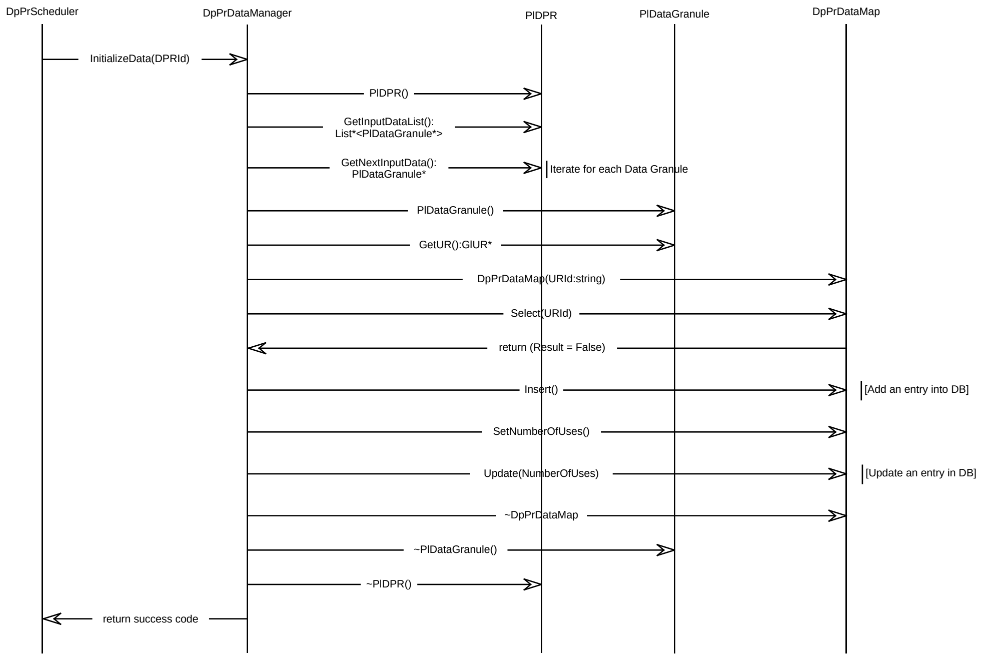


Figure 4.5-9. DpPrDataMap Initialization

4.5.3.2 Local Data Management

4.5.3.2.1 Abstract

The Processing CSCI has the responsibility to ensure that all input data are available and reside on the science processing hardware resources before the execution of PGEs. The Processing CSCI allocates sufficient resources to support output data that will be produced by a PGE even before ensuring that all input data are local. If our science Processing hardware resource can accommodate all output data, then it proceeds with determining the availability of input data. To manage this task, the Processing CSCI must determine if data is located locally on the platform that the PGE is using for execution, or on some other local science Processing hardware resource, or data is not available on our science Processing hardware resource at all. If the data resides on this platform, no further actions need to be done. If the location of data is on some other local science Processing hardware resource, then a location is allocated by Resource Management and the data will be copied from some other resource platform to this platform at the new location provided by Resource Management. This new location and other characteristics of this data are stored in PDPS Database. In the case that the data is not available on our science Processing hardware resource at all, the Resource Management allocates space for this data on this platform. The processing CSCI then builds a Data Server request including ACQUIRE command(s) that will be used to request the staging of data to a science processing resource. Also this new location provided by Resource Management and other characteristics of this data are stored in PDPS Database. The localization of the data occurs when the ReleaseDPRjob operation of DpPrScheduler class is activated.

4.5.3.2.2 Stimulus

When the Planning CSCI releases a Data Processing Request to AutoSys' daily job schedule, AutoSys will kick off a job which invokes MakeDataLocal of DpPrDataManager class.

4.5.3.2.3 Desired Response

This activity will result in the initiation of a persistent data structure used to retain knowledge about the location of data on the Science Processing Hardware resources.

4.5.3.2.4 Participating Classes From the Object Model

- COTS
- DpPrDataManager
- DpPrResourceManager
- PlDPR
- DpPrDataMap
- PlDataGranule

4.5.3.2.5 Scenario Description

a) When the ReleaseDPRJob operation is initiated on the DpPrScheduler, the following steps will occur:

    1) The AutoSys will kick off a job that invokes MakeDataLocal operation of DpPrDataManager class to perform the following steps for each job:

        (a) Iterate through the output data granule list, call DpPrResourceManager to allocate resource for each output data.

        (b) If all output data granules are allocated successfully, iterate through the input data granule list, determine if data is available at science processing hardware resource or not.

        (c) If data is resident on this local science processing hardware resource, the PDPS Database entry will be updated.

        (d) If data is not resident on this local science processing hardware resource, but is located on some other local science processing hardware resource, then ask DpPrResourceManager to allocate space for this data on this local resource and copy data from nearby resource to this local resource at the new location. Go to PDPS Database and add an entry for this data with new location and other characteristic information.

        (e) If data is not located anywhere on our local science processing hardware resource, then ask DpPrResourceManager to allocate space for this data on this local resource, and request Data Server to stage data to our local science processing hardware resource (see Data Staging section). Go to PDPS

Database and add an entry for this data with new location, status is set to STAGING, and other characteristic information.

b) END OF SCENARIO.

4.5.3.2.6 Event Trace

Figures 4.5-10 through 4.5-12 show the Local Data Management.

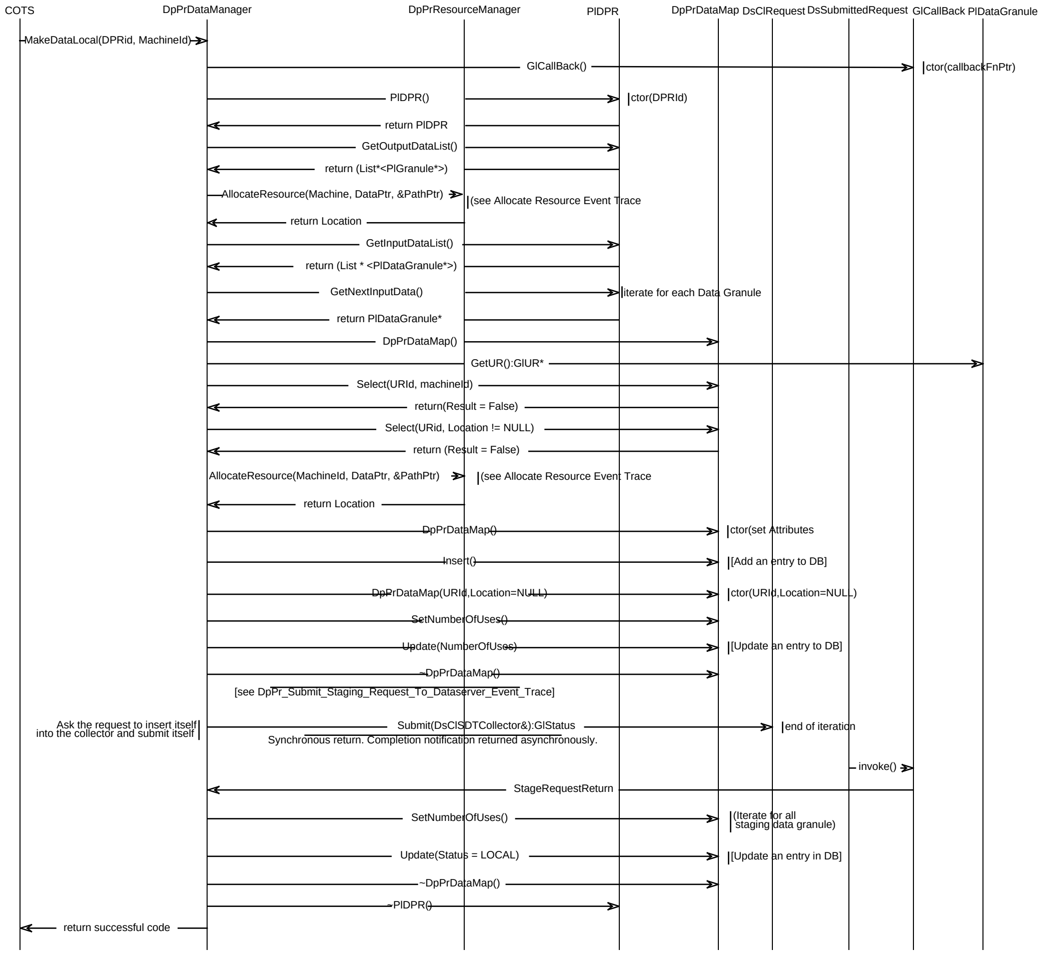


Figure 4.5-10. Local Data Management (Data Staging Required)

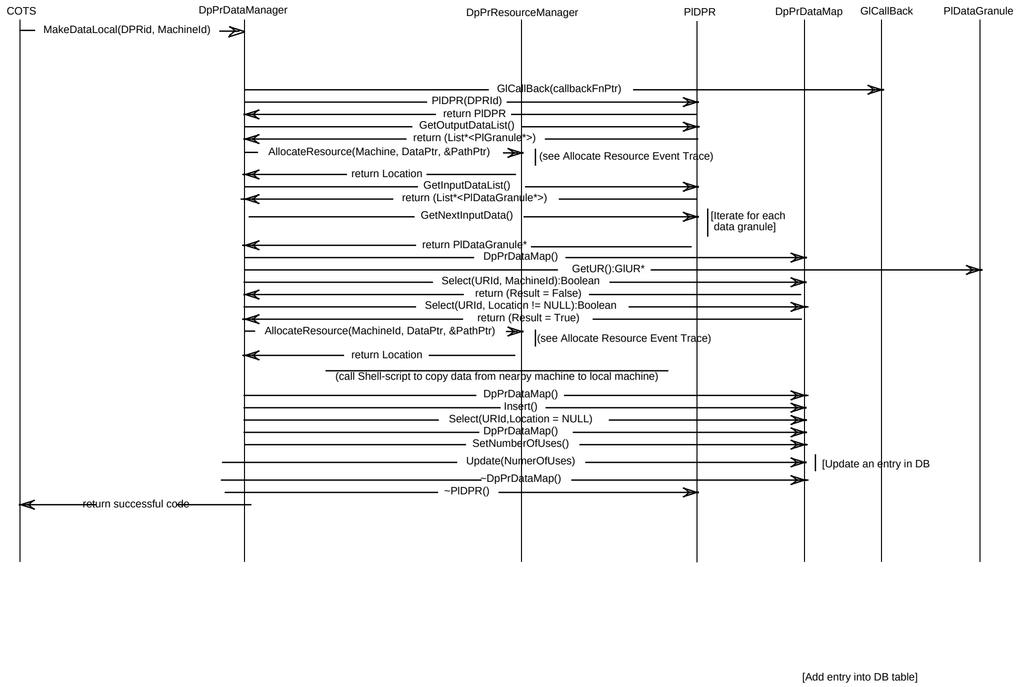

COTS
DpPrDataManager
DpPrResourceManager
PlDPR
DpPrDataMap
GlCallBack
PlDataGranule
MakeDataLocal(DPRid, MachineId)
GlCallBack(callbackFnPtr)
PlDPR(DPRId)
return PlDPR
GetOutputDataList()
return (List*<PlGranule*>)
AllocateResource(Machine, DataPtr, &PathPtr)
(see Allocate Resource Event Trace)
return Location
GetInputDataList()
return (List*<PlDataGranule*>)
GetNextInputData()
[Iterate for each data granule]
return PlDataGranule*
DpPrDataMap()
GetUR():GlUR*
Select(URId, MachineId):Boolean
return (Result = False)
Select(URId, Location != NULL):Boolean
return (Result = True)
AllocateResource(MachineId, DataPtr, &PathPtr)
(see Allocate Resource Event Trace)
return Location
(call Shell-script to copy data from nearby machine to local machine)
DpPrDataMap()
Insert()
Select(URId,Location = NULL)
DpPrDataMap()
SetNumberOfUses()
Update(NumerOfUses)
[Update an entry in DB
~DpPrDataMap()
~PlDPR()
return successful code
[Add entry into DB table]

Figure 4.5-11. Local Data Management (Local Data Movement)

4.5.3.3 Data Staging

4.5.3.3.1 Abstract

Data Staging is an internally generated Data Processing process which must occur prior to PGE execution. Data Staging will be initiated when the COTS has initiated the Data Management Job associated with the AutoSys job box created from a Data Processing Request. After determining that there are sufficient resources required to support the execution of the PGE, the Data Management services will initiate the data staging process. The Data Management services will use Science Data Server CSCI provided public classes to request data staging. The Data Staging process is the series of steps followed to transfer data from the Data Server subsystem to the Data Processing subsystem. The staging of all data required by a PGE is completed prior to the execution of a PGE.

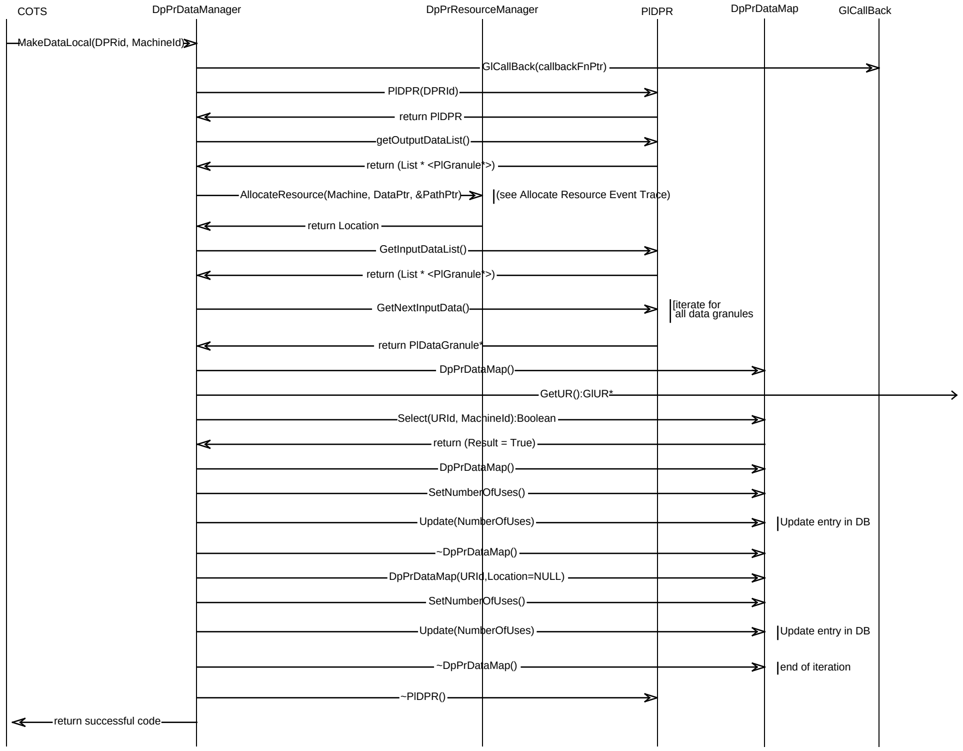


Figure 4.5-12. Local Data Management (Data Resides on Science Processing Resource)

4.5.3.3.2 Stimulus

As the jobs which are part of the job box associated with a Data Processing Request are executed, one of the jobs which performs Data Management services will request data staging to be initiated upon determining that sufficient resources are available to support data staging and PGE execution.

4.5.3.3.3 Desired Response

The data staging process will be initiated by issuing a request to the Science Data Server CSCI. Part of this request will consist of an Acquire command which will be used to request the staging of data to a science processing resource. Also provided with this request is a Callback Mechanism which is provided by the Processing CSCI to the Science Data Server CSCI to return the success or failure status of the staging operations.

4.5.3.3.4 Participating Classes From the Object Model

- DpPrDataManager
- DsClESDTReferenceCollector
- DsClRequest

- DsClCommand
- PlDPR
- PlDataGranule
- GlCallBack

4.5.3.3.5 Scenario Description

a. When AutoSys determines that a Data Management job can be initiated, i.e., all the job dependencies defined for this job have been fulfilled, the Data Management Job is executed and the following activities occur:

   1. Build a request to provide to the Science Data Server CSCI. This request will consist of a series of commands which will be the Acquire commands needed to stage all of the data required for the PGE associated with this Data Processing Request. The Acquire commands are actually obtained from the PDPS Database and added to the Science Data Server CSCI public class. After completing the building of the request, the request is submitted to the Science Data Server CSCI. The Data Management Job will wait until a notification is received back from the Science Data Server CSCI that this request was fulfilled successfully or unsuccessfully.
   2. When the Science Data Server CSCI activates the Callback mechanism to inform the Processing CSCI of the success or failure of the request, the Data Management Job will map the success or failure of the data staging operation to a return status code and gracefully terminate. The return status code will be used by AutoSys to determine the success or failure of this job. If the return code indicates success, AutoSys will initiate the next dependent job, i.e., the PGE Preparation Job. If the return code indicates failure, AutoSys will take appropriate error recovery activities, such as alerting this operations staff. More information on the failure of data staging contained in the Section 4.5.4, Failure of Data Staging scenario.
   3. Logging of all information related to these events will occur through AutoSys and MSS provided services used by the Data Management component of the Processing CSCI.

b. End of Scenario.

4.5.3.3.6 Event Trace

Figure 4.5-13 shows the data staging event trace.

4.5.3.4 Failure of Data Staging

4.5.3.4.1 Abstract

The failure of data staging can occur for numerous reasons which are internal or external to the domain of the Processing CSCI. In all cases, the Science Data Server CSCI will inform the Processing CSCI of the failure of data staging. If the cause of the problem is within the science processing resources, appropriate error recovery actions will be initiated which may lead to the re-initiation of staging, informing Operations and awaiting manual intervention, or taking appropriate termination activities.

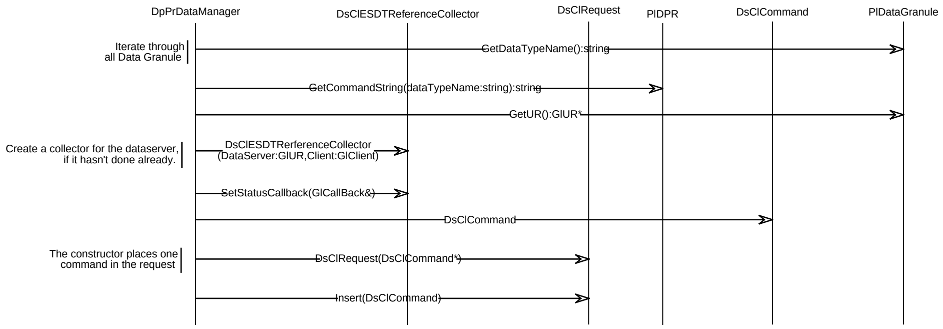

DpPrDataManager
DsClESDTReferenceCollector
DsClRequest
PlDPR
DsClCommand
PlDataGranule
Iterate through all Data Granule
GetDataTypeName():string
GetCommandString(dataTypeName:string):string
GetUR():GlUR*
Create a collector for the dataserver, if it hasn't done already.
DsClESDTRerferenceCollector (DataServer:GlUR,Client:GlClient)
SetStatusCallback(GlCallBack&)
DsClCommand
The constructor places one command in the request
DsClRequest(DsClCommand*)
Insert(DsClCommand)

Figure 4.5-13. Data Staging Event Trace

4.5.3.4.2 Stimulus

The stimulus which triggers a Data Staging Failure occurs during the interaction of the Data Server with the staging resources and is therefore an external event. This scenario will address the outcome on Processing due to such an event. Using a callback mechanism which is provided by the Processing CSCI, the Science Data Server CSCI will inform the Processing CSCI of the failure of data staging. This failure information will be transferred into a status return code which is used to inform AutoSys of the failure of the Data Management job.

4.5.3.4.3 Desired Response

When alerted to the failure of the Data Management Job through the status return code mechanism, AutoSys will activate appropriate error recovery actions. Initially, this amounts to alerting the Operations staff, logging information about the failure to AutoSys job log as well as the system log retained by MSS.

4.5.3.4.4 Participating Classes from the Object Model

- GlCallBack
- DpPrDataManager
- COTS

4.5.3.4.5 Scenario Description

a. When the Science Data Server CSCI activates the Callback mechanism to inform the Processing CSCI of the success or failure of the request, the Data Management Job will map the success or failure of the data staging operation to a return status code and gracefully terminate. The return status code will be used by AutoSys to determine the success or failure of this job. If the status return code indicates failure, AutoSys will take appropriate error recovery activities, such as the following:

    1. Activate the alert mechanisms to inform the operations staff.

2. Check the local resources for problems which may have caused the staging failure.
3. Log fault information to MSS.
4. Delete the input data that has been staged thus far, unless subsequent, near-term, processing requires it.
5. Update the operational mode of the failed resource to "Off-Line" so that these resources do not get allocated for subsequent processing until the resource problem is corrected.
6. Deallocate the remaining resources initially allocated for this run of the PGE.

b. End of Scenario.

4.5.3.4.6 Event Trace

Figure 4.5-14 shows the failure of data staging event trace.

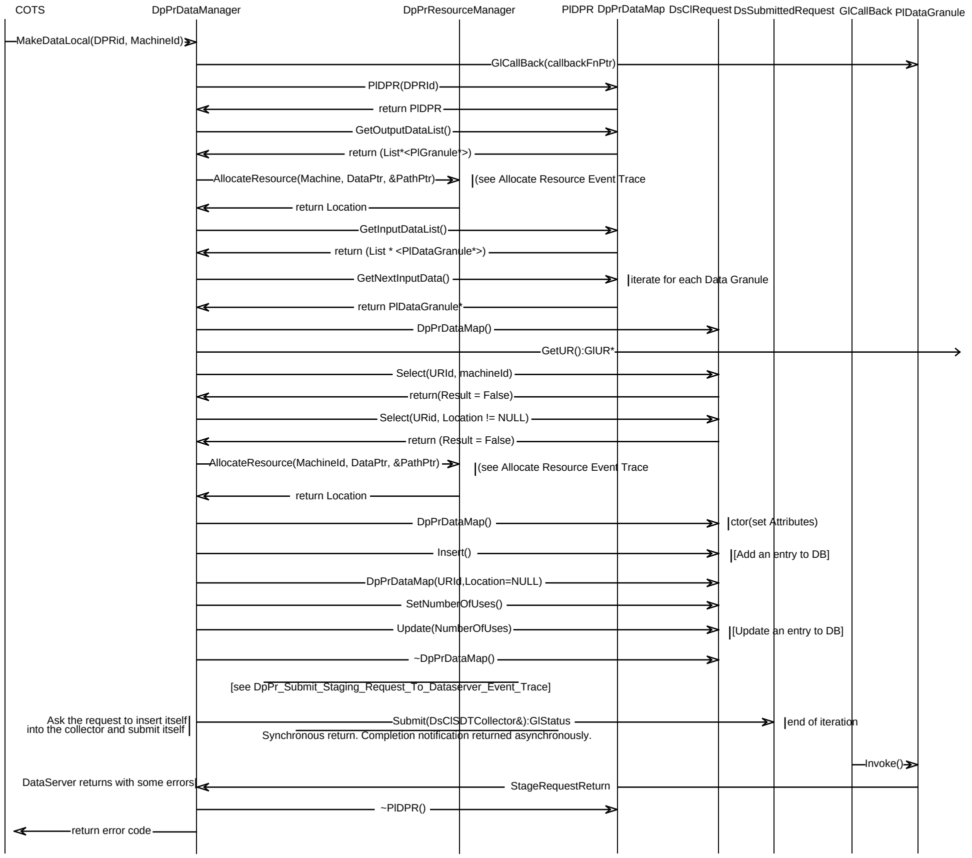

COTS
DpPrDataManager
DpPrResourceManager
PlDPR
DpPrDataMap
DsClRequest
DsSubmittedRequest
GlCallBack
PlDataGranule
MakeDataLocal(DPRid, MachineId)
GlCallBack(callbackFnPtr)
PlDPR(DPRId)
return PlDPR
GetOutputDataList()
return (List*<PlGranule*>)
AllocateResource(Machine, DataPtr, &PathPtr)
(see Allocate Resource Event Trace
return Location
GetInputDataList()
return (List * <PlDataGranule*>)
GetNextInputData()
iterate for each Data Granule
return PlDataGranule*
DpPrDataMap()
GetUR():GlUR*
Select(URId, machineId)
return(Result = False)
Select(URid, Location != NULL)
return (Result = False)
AllocateResource(MachineId, DataPtr, &PathPtr)
(see Allocate Resource Event Trace
return Location
DpPrDataMap()
ctor(set Attributes)
Insert()
[Add an entry to DB]
DpPrDataMap(URId,Location=NULL)
SetNumberOfUses()
Update(NumberOfUses)
[Update an entry to DB]
~DpPrDataMap()
[see DpPr_Submit_Staging_Request_To_Dataserver_Event_Trace]
Ask the request to insert itself
into the collector and submit itself
Submit(DsClSDTCollector&):GlStatus
end of iteration
Synchronous return. Completion notification returned asynchronously.
Invoke()
DataServer returns with some errors
StageRequestReturn
~PlDPR()
return error code

Figure 4.5-14. Failure of Data Staging Event Trace

4.5.3.5 Data Destaging

4.5.3.5.1 Abstract

Data Destaging is an internally generated Data Processing process which must occur after completion of PGE execution. Data Destaging will be initiated when the COTS has initiated the PGE Post-Processing Job associated with the AutoSys job box created from a Data Processing Request. The Data Management services will used Science Data Server CSCI provided public classes to request data destaging. The Data Destaging process is the series of steps followed to transfer data from the Processing CSCI to the Science Data Server CSCI. The destaging of all output data produced by a PGE is completed after the execution of a PGE.

4.5.3.5.2 Stimulus

As the jobs which are part of the job box associated with a Data Processing Request are executed, the PGE Post-Processing job will request data destaging to be initiated.

4.5.3.5.3 Desired Response

The data destaging process will be initiated by issuing a request to the Science Data Server CSCI. Part of this request will consist of an Insert command which will be used to request the destaging of data to a science processing resource. Also provided with this request is a Callback Mechanism which is provided by the Processing CSCI to the Science Data Server CSCI to return the success or failure status of the destaging operations.

4.5.3.5.4 Participating Classes From the Object Model

- COTS
- DpPrDataManager
- DpPrResourceManager
- DsClESDTReferenceCollector
- DsClRequest
- DsClCommand
- PlDPR
- PlDataGranule
- GlCallBack

4.5.3.5.5Scenario Description

a. When AutoSys determines that a PGE Post-Processing job can be initiated, i.e., all the job dependencies defined for this job have been fulfilled, the PGE Post-Processing Job is executed and the following activities occur:

    1. Build a request to provide to the Science Data Server CSCI. This request will consist of a series of commands which will be the Insert commands needed to destage all of the data required for the PGE associated with this Data Processing Request. The Insert commands are actually obtained from the PDPS Database and added to the Science Data Server CSCI public class. After completing the building of the request,

the request is submitted to the Science Data Server CSCI. The Data Management Job will wait until a notification is received back from the Science Data Server CSCI that this request was fulfilled successfully or unsuccessfully.

2. When the Science Data Server CSCI activates the Callback mechanism to inform the Processing CSCI of the success or failure of the request, the PGE Post-Processing Job will map the success or failure of the data destaging operation to a return status code and gracefully terminate. The return status code will be used by AutoSys to determine the success or failure of this job. If the return code indicates success, AutoSys will conclude the processing for this job and move on to initiate the next job awaiting execution. If the return code indicates failure, AutoSys will take appropriate error recovery activities, such as alerting this operations staff. More information on the failure of data staging contained in the Section 4.5.6, Failure of Data Destaging scenario, or Section 4.5.7, Failure of Data Server Communication scenario.
3. Logging of all information related to these events will occur through AutoSys and MSS provided services used by the Data Management component of the Processing CSCI.

b. End of Scenario.

4.5.3.5.6 Event Trace

Figure 4.5-15 shows the deallocate data event trace. Figure 4.5-16 shows the data destaging event trace.

4.5.3.6 Failure of Destaging

4.5.3.6.1 Abstract

The failure of data staging or destaging can occur for numerous reasons which are internal or external to the domain of the Processing CSCI. In all cases, the Science Data Server CSCI will inform the Processing CSCI of the failure situation. If the cause of the problem is within the science processing resources, appropriate error recovery actions will be initiated which may lead to the re-initiation of staging or destaging, informing Operations and awaiting manual intervention, or taking appropriate termination activities.

4.5.3.6.2 Stimulus

The stimulus which triggers a data staging or destaging failure occurs during the interaction of the Science Data Server CSCI with the staging resources and is therefore an external event. This scenario will address the outcome on the Processing CSCI due to such an event. Using a callback mechanism which is provided by the Processing CSCI, the Science Data Server CSCI will inform the Processing CSCI of the failure of data staging or destaging. This failure information will be transferred into a status return code which is used to inform AutoSys of the failure of the PGE or Post-Processing job.

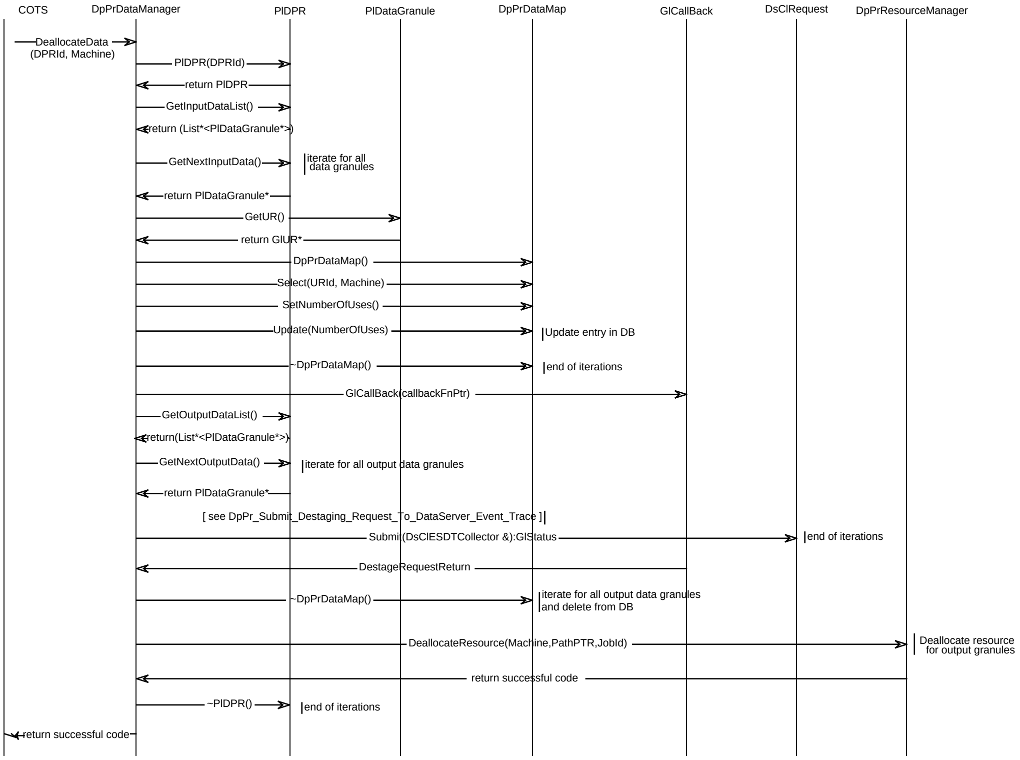


Figure 4.5-15. Deallocate Data Event Trace

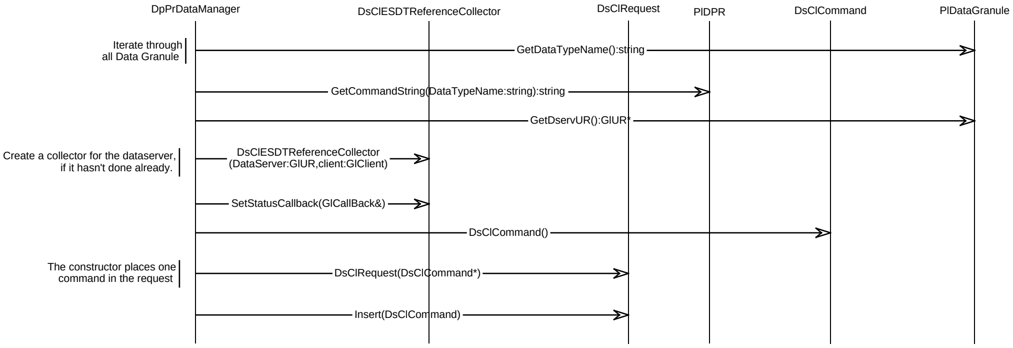

DpPrDataManager
DsClESDTReferenceCollector
DsClRequest
PlDPR
DsClCommand
PlDataGranule
Iterate through
all Data Granule
GetDataTypeName():string
GetCommandString(DataTypeName:string):string
GetDservUR():GlUR*
Create a collector for the dataserver,
if it hasn't done already.
DsClESDTReferenceCollector
(DataServer:GlUR,client:GlClient)
SetStatusCallback(GlCallBack&)
DsClCommand()
The constructor places one
command in the request
DsClRequest(DsClCommand*)
Insert(DsClCommand)

Figure 4.5-16. Data Destaging Event Trace

4.5.3.6.3 Desired Response

When alerted to the failure of the PGE Post-Processing Job through the status return code mechanism, AutoSys will activate appropriate error recovery actions. Initially, this amounts to alerting the Operations staff, logging information about the failure to AutoSys job log as well as the system log retained by MSS.

On failing to destage all of the output data produced by the execution of the current PGE, the Operations should attempt to determine if the problem lies with the local resources and if so, should take steps to identify the failed resources in order to prevent similar problems during subsequent processing. In any event, deallocation of resources should not occur for this process until the data are recovered and destaging has been performed.

4.5.3.6.4 Participating Classes from the Object Model

- DpPrDataManager
- GlCallBack
- COTS

4.5.3.6.5 Scenario Description

a. When the Science Data Server CSCI activates the Callback mechanism to inform the Processing CSCI of the success or failure of the request, the Data Management Job will map the success or failure of the data destaging operation to a return status code and gracefully terminate. The return status code will be used by AutoSys to determine the success or failure of this job. If the status return code indicates failure, AutoSys will take appropriate error recovery activities, such as the following:

    1. Activate the alert mechanisms to inform the operations staff.
    2. Check the local resources for problems which may have caused the destaging failure.
    3. Log fault information to MSS.
    4. Update the operational mode of the failed resource to "Off-Line" so that these resources do not get allocated for subsequent processing until the resource problem is corrected and the generated outputs of the PGE are destaged successfully.

b. End of Scenario.

4.5.3.6.6 Event Trace

Figure 4.5-17 shows the failure of a data destaging event trace.

4.5.3.7 Failure of Data Server Communication

4.5.3.7.1 Abstract

The failure of Data Server Communication during staging or destaging of data is handled similarly to the handling of the failure of staging. All appropriate error recovery actions will be tried including initiating the data staging or destaging again, suspending further processing of the PGE job definition (i.e., Data Processing Request) and awaiting Operations intervention, or possibly terminating further processing of the PGE job definition and submitting a communications fault to MSS.

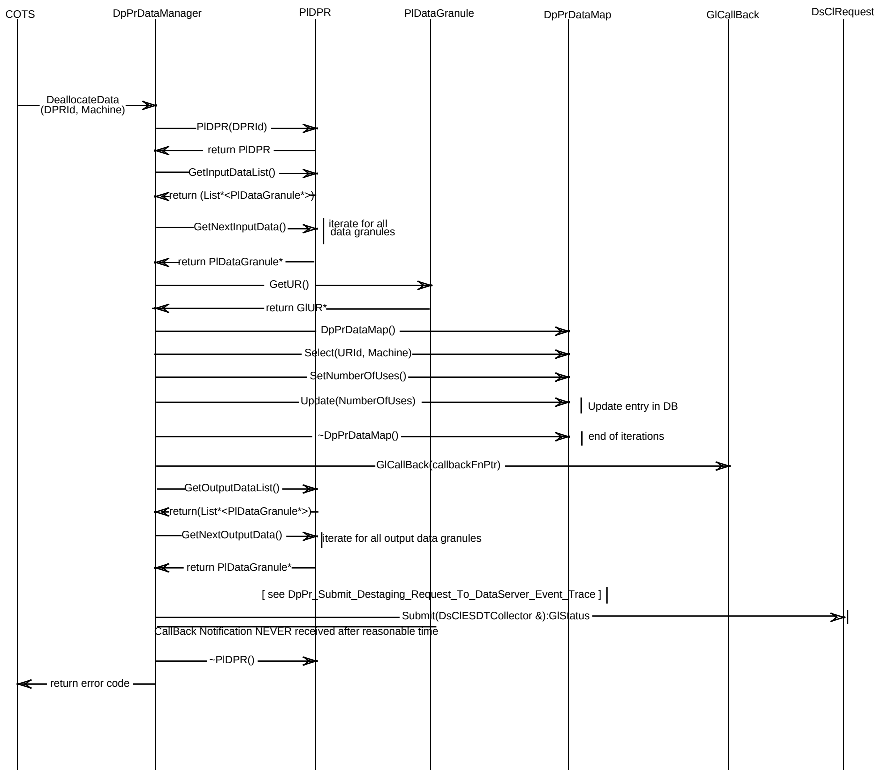


Figure 4.5-17. Failure of a Data Destaging Event Trace

4.5.3.7.2 Stimulus

The stimulus which triggers a Data Server Communication Failure occurs external to the Processing CSCI. This scenario will address the outcome on Processing when such an event occurs during Process Control File (PCF) destaging.

4.5.3.7.3 Desired Response

A single Process Control File (PCF) is created by the PGE Execution Management services and may undergo minor modifications during the execution of the current PGE. On failing to destage the Process Control File (PCF) due to a failure of the communication channels between Processing and Data Server, the Execution Management services should be able to determine if the problem lies with the local resources. If so, steps should be taken to identify the failed resources in order to prevent similar problems during subsequent processing. However, since the PCF may still remain on the failed resource, if deallocation cannot be achieved after several attempts, the resource is taken off-line to allow for operator intervention. In any event, deallocation of resources should not occur for this process until the PCF is recovered and destaging has completed successfully.

4.5.3.7.4 Participating Classes from the Object Model

- COTS
- DpPrPge
- DpPrExecutionManagement
- DpPrResourceManagement
- PlDataGranule
- PlDpr
- DsClESDTReferenceCollector
- DsClRequest
- DsClCommand
- MsMgCallBacks
- MsManager
- MsEvent

4.5.3.7.5 Scenario Description

a. After the COTS Scheduler has activated the destaging of the Process Control File (PCF) to support post-processing of a PGE, the following activities may occur:

   1. With the data successfully copied to the Data Server, deallocation of Processing resources may proceed.
   2. The Execution Management service issues a request to destage the Process Control File (PCF) to support the SCF analysis of a failed, or suspect processing attempt.
   3. If an event, which is external to the Processing CSCI, causes the communication with the Data Server to become disrupted, the Data Server will not be able to fulfill the destaging request. The end result is that the Execution Management "callback" routine will not be activated by the Data Server and Execution Management, therefore, will not receive a complete notification.
   4. If a problem occurs during the destaging effort, the Execution Management destaging services will time-out and conclude that the destage operation has failed. Resource Management services will then be called upon to perform a check on the local resources for problems which may have caused the destaging failure.
   5. Deallocate the remaining processing resources initially allocated for this run of the PGE, leaving allocated those resources where output data may still reside.
   6. If it is discovered that local resources were the cause of the destaging failure, a resource fault will be logged with MSS, otherwise a communications fault will be logged.
   7. Update the COTS Scheduler interface to indicate a failure during Post-processing.

b. End of Scenario.

4.5.3.7.6 Event Trace

Figure 4.5-18 shows the failure of data server communication event trace.

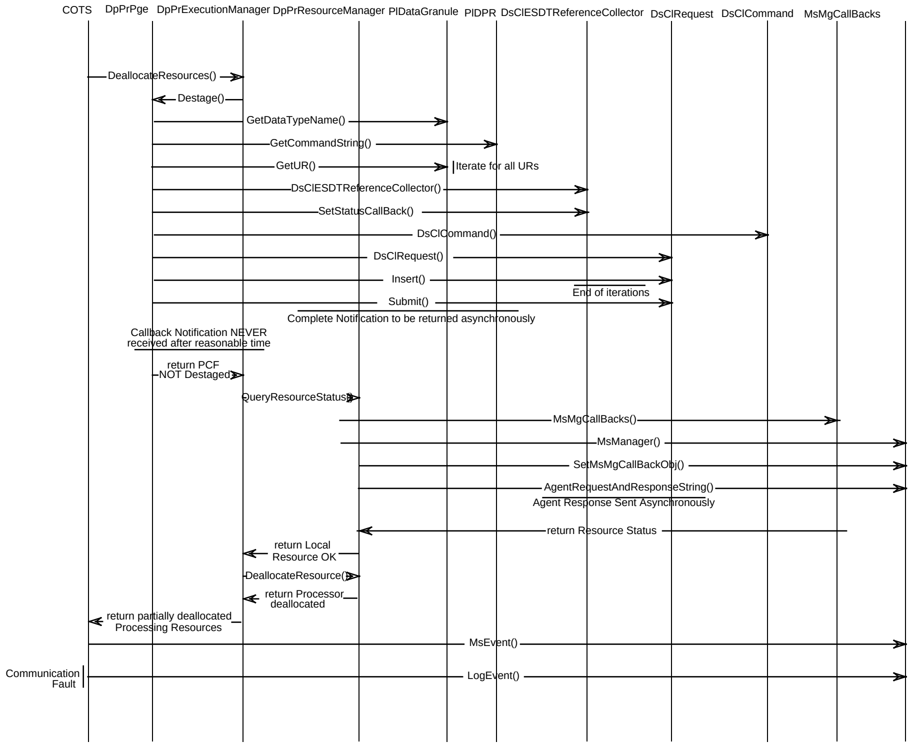


Figure 4.5-18. Failure of Data Server Communication Event Trace

4.5.4 Execution Management Scenarios

Execution Management services are used to initiate and monitor the execution of a PGE. All support for PGE execution, i.e., SDP Toolkit interfaces, production history generation, resource monitoring, etc., will be provided by these services. The execution of a PGE is performed on Data Processing subsystem's resources and is initiated by the Processing CSCI.

The execution of a PGE is initiated when the following conditions are met:

a. Data Processing subsystem resources are available to support the successful execution of the PGE.

b. The priority and resource information assigned to the Data Processing Request has positioned the Data Processing Request, within the COTS Scheduler, as the next request to execute.

Conceptually, a PGE is defined as any processing job that requires Data Processing subsystem resources. Therefore, a PGE can define different types of processing; science software, quality assurance, or science pre-processing. If there are standard types of processing jobs that are performed periodically, these jobs are planned and submitted to Processing from Planning. Of course, there may be exceptions, such as operations staff intervention in processing of a PGE, where this is not followed. Most PGEs will be defined as science software PGEs.

The execution of a science software PGE will result in the generation of data products. These data products are of two types:

a. Intermediate—This data is used to support the execution of other PGEs. Even though this data may only be required for a finite period of time, to facilitate the generation of relatively long term products, all such data files will be archived at the Data Server.

b. Final Data Product—This data is defined as Level 1, Level 2, Level 3, or Level 4 data products. These products can be defined as Intermediate data to support the production of other products. For example, a Level 1 product is used to create a Level 2 product. The Level 2 product is a Final Data Product, but could also be used to create a Level 3 product.

Before initiating the execution of the PGE, the Processing CSCI provides information to the PGE about the location and names of input data and the destination and names of output data. Currently, this information is provided through a data mapping mechanism known as a Process Control File (PCF). This file contains information linking the logical representation of data to its physical counterpart, i.e., logical unit numbers to file paths. This data mapping represents the only interface between the Processing CSCI and the SDP Toolkit API. As such, this interface will be used to provide metadata to the PGE, through the SDP Toolkit, to be associated with the data products.

During execution, the Processing CSCI performs a monitoring role. Periodically, the Processing CSCI collects information on executing PGE(s). This information includes execution errors and warnings generated from the science software, as well as certain runtime performance statistics. Also, the Processing CSCI monitors the resources being used by the PGE. By comparing current resource usage data with the nominal resource usage data prepared by the Algorithm Integration and Test Team, the Processing CSCI can determine if a resource fault exists, or that a PGE fault has occurred. The Processing CSCI may terminate a PGE if such runtime information indicates that the PGE is not performing within the nominal bounds defined at AI&T, or if resource faults are occurring. To support the monitoring of the PGE, the Processing CSCI will use services provided by MSS.

Currently, no unique services have been identified to provide support for the PGEs which are not science software.

Figure 4.5-19 shows a state transition diagram for the execution of a PGE. This diagram should be reviewed while reading the scenarios and event traces.

The scenarios provided for Execution Management are defined as the following

a. Initiate Execution—This scenario provides information on the required activities for initiation of the execution of a PGE. See scenario "Execution Management: Initiate PGE Execution" for more information.

b. Monitor Execution—These scenarios provides information on the required activities to monitor PGE execution. See scenarios "Execution Management: Monitor Resource", "Execution Management: Monitor Performance," and "Execution Management: Monitor Status Return" for more information.

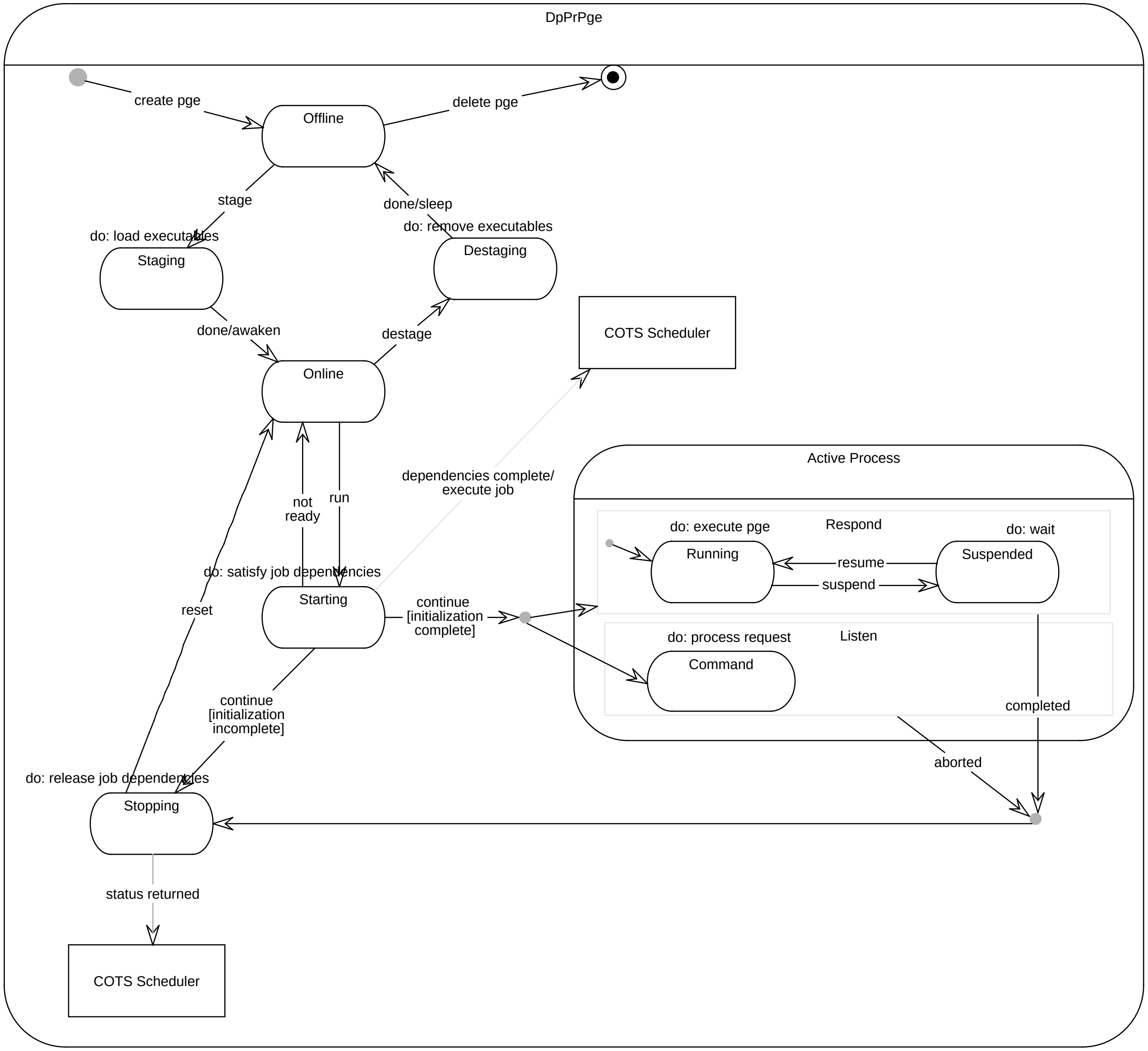


Figure 4.5-19. PGE State Transition Diagram

c. Execution post-processing—This scenario provides information on the required activities for PGE execution post-processing. See scenario "Execution Management: PGE Execution Post-Processing" for more information.

d. Failure of Execution—This scenario provides information on the required activities following failure of PGE execution. See scenario "Execution Management: Failure of PGE Execution" for more information.

e. Failure of Data Server Communications—This scenario provides information on the required activities following failure of communications during Data Server contact. See scenario "Execution Management: Failure of Data Server Communications" for more information.

f. Failure of Processing Resource—This scenario provides information on the required activities following failure of a processing resource. See scenario "Execution Management: Failure of Processing Resource" for more information.

4.5.4.1 Initiate Execution

4.5.4.1.1 Abstract

When it is determined by the COTS Scheduler that the processing of a PGE can proceed, PGE preparation activities can begin. These activities consist of resource allocation and SDP Toolkit initialization.

Currently, the SDP Toolkit requires the creation of a Process Control File (or PCF), which acts as the interface between the science algorithm and the Processing CSCI. The data mappings contained within the PCF contain all of the information needed for the SDP Toolkit to access the data requested by the science algorithm. Since only one Process Control File is associated with a given PGE, it must be created and populated with new mapping information for each unique run of the PGE.

In addition to the input data that is staged for the PGE, SDP Toolkit and algorithm specific Status Message Files (SMFs) must also be staged to provide for the error and status reporting needs of the PGE. However, before any data item can be staged, the necessary disk resources must first be marshaled for the current PGE. This occurs through an allocation process which is managed by the Resource Management services.

4.5.4.1.2 Stimulus

The stimulus to initiate PGE execution is received as a request from the COTS scheduler package. The events which occur just prior to the start of PGE execution signal the completion of resource allocation, data staging and SDP Toolkit initialization for this task.

4.5.4.1.3 Desired Response

The execution of a PGE job will be triggered by the COTS scheduling package.

4.5.4.1.4 Participating Classes from the Object Model

- COTS
- DpPrExecutable

- DpPrPcf
- DpPrPge
- DpPrExecutionManager
- DpPrResourceManager
- PlDataGranule
- PlDpr
- DsClESDTReferenceCollector
- DsClRequest
- DsClCommand

4.5.4.1.5 Scenario Description

a. When the COTS scheduler has requested the initialization of a PGE, the following activities are done:

   1. Perform the resource preparation activities required to support the staging of data and PGE, as well as the processing of the PGE (e.g., allocate disk space, CPU(s)).
   2. If not already resident on the local host, stage the actual PGE executables and associated Status Message Files (SMFs) which were delivered along with the science algorithm.
   3. Construct the Process Control File, for this instance of the PGE, to create the mapping of logical identifiers to physical file and runtime parameter values.
   4. Update the Process Control Table with "live" information to complete the mapping of logical identifiers to physical file and parameter values.
   5. Send Event to the COTS scheduler signifying the completion of the initialization phase.
   6. Log the success or failure to initialize the execution phase of the PGE execution. (i.e., job status is "Success," or "Failure").
   7. With all job dependencies satisfied, the COTS scheduler issues a start processing request to begin the actual PGE job execution.

b. End of "Execution Management (Initiate Execution)".

4.5.4.1.6 Event Trace

Figure 4.5-20 shows the initiate execution event trace diagram.

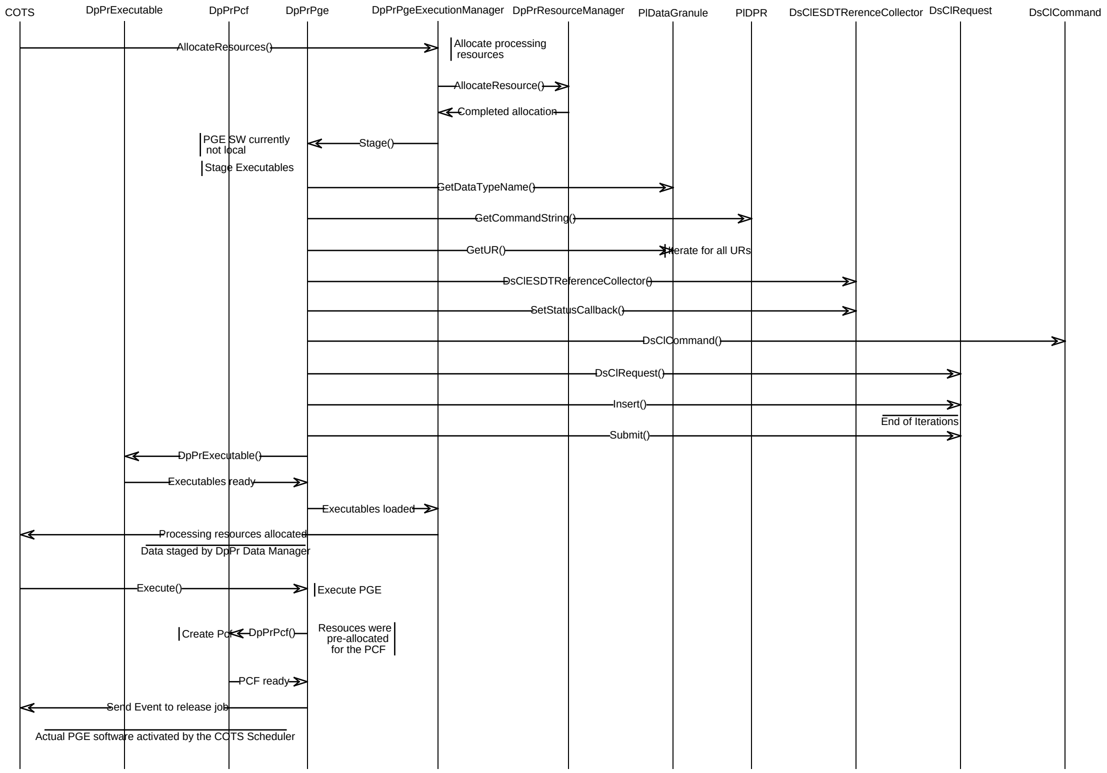


Figure 4.5-20. Initiate Execution Event Trace

4.5.4.2 Monitor Execution

4.5.4.2.1 Abstract

Execution Management services will monitor the execution of a PGE and the resources that the PGE is using during execution. To monitor the execution of the PGE at runtime, the Execution Management services will obtain performance status information through a MSS defined method. To obtain the termination status of a PGE, the final state of processing will be returned to the COTS Scheduler upon completion of the PGE's execution.

The definition of the status condition codes used by the PGEs will be determined through ECS interaction with Instrument Team algorithm developers. Once a complete set of status condition codes is defined and distributed to Instrument Team sites, all science algorithm developers will be required to return one of these condition codes at the terminus of their processing. This will further require science algorithm developers to handle error conditions gracefully to ensure that this critical section of code always gets invoked.

Resource monitoring will consist of the monitoring of the usage of resources, e.g., CPU, disk space, and memory by the PGE. During AI&T, the PGE resource usage data will be established. These data are likely to include averages and maximums of the CPU, memory, and disk space that a PGE uses during execution. This information will be updated as the PGE is used in the production environment. During execution, resource monitoring will alert operations if the PGE has reached some percentage over the average, or maximum allowed. Resource monitoring also will continuously check on the health of the resources being used. Resources will be monitored jointly by MSS and Processing.

All of the information acquired through monitoring will be captured through the COTS logging mechanisms.

4.5.4.2.2 Stimulus

The Execution Management services will perform these monitoring activities on periodic basis for selected PGE runs.

4.5.4.2.3 Desired Response

The execution of a PGE will be monitored to check for anomalies which might develop with the resources, the performance of the algorithm, or with the algorithm itself. If conditions warrant, the execution of the PGE may be terminated by operations personnel.

4.5.4.2.4 Participating Classes from the Object Model

- COTS
- DpPrPge
- DpPrExecutable
- DpPrExecutionManager
- DpPrResourceManager
- DpPrComputer
- MsMgCallBacks
- MsManager
- MsEvent

4.5.4.2.5 Scenario Description

a. During PGE execution, Execution Management Services will do the following:

Case 1: Monitor Resource Health during PGE execution

1. Check the health of the resources being used by the PGE. This is an MSS provided service.
2. If the state of the resources being used by the currently executing PGE are failing or have failed, terminate execution of the PGE gracefully.
3. The COTS Scheduler log is updated with the final state of the PGE.
4. An exception record has been submitted to the MSS Event log to indicate that the PGE has been aborted.

Case 2: Monitor performance during PGE execution

1. Check on the performance of the currently executing PGE.

2. If the usage of the resources is not following the established performance characteristics for the PGE, terminate execution of the PGE gracefully.
3. The COTS Scheduler log is updated with the final state of the PGE.
4. An exception record has been submitted to the MSS Event log to indicate that the PGE has been aborted.

Case 3: Monitor status return following PGE execution

1. The executing PGE continually updates the Toolkit Status Message Log with algorithm, or SDP Toolkit specific status information.
2. The PGE, upon detecting a fatal error condition, updates the Toolkit Status Message Log and exits gracefully, returning a predefined condition code to the COTS Scheduler.
3. This information is captured by the COTS Scheduler to determine the reason for the PGE failure.
4. The COTS Scheduler log is updated with the final state of the PGE.
5. An exception record has been submitted to the MSS Event log to indicate that the PGE has been aborted.

b. End of "Execution Management (Monitor Execution)."

4.5.4.2.6 Event Trace

Figure 4.5-21 shows the Case 1 monitor resource health trace. Figure 4.5-22 shows the Case 2 monitor performance trace. Figure 4.5-23 shows the Case 3 monitor status return trace.

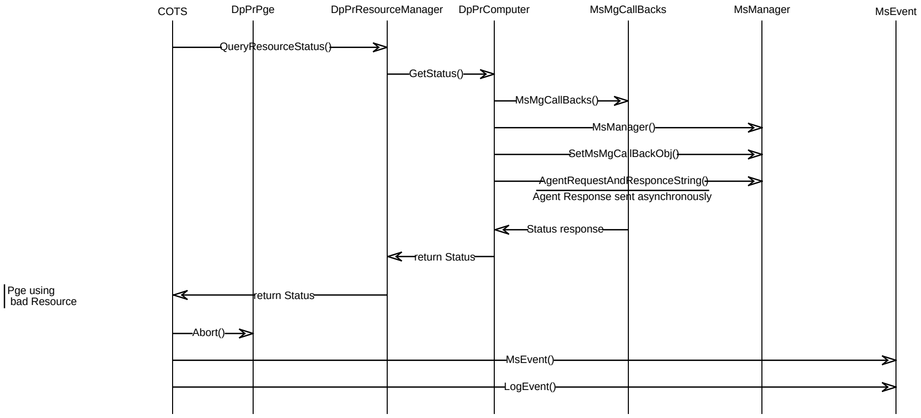


Figure 4.5-21. Case 1: Monitor Resource Health Trace

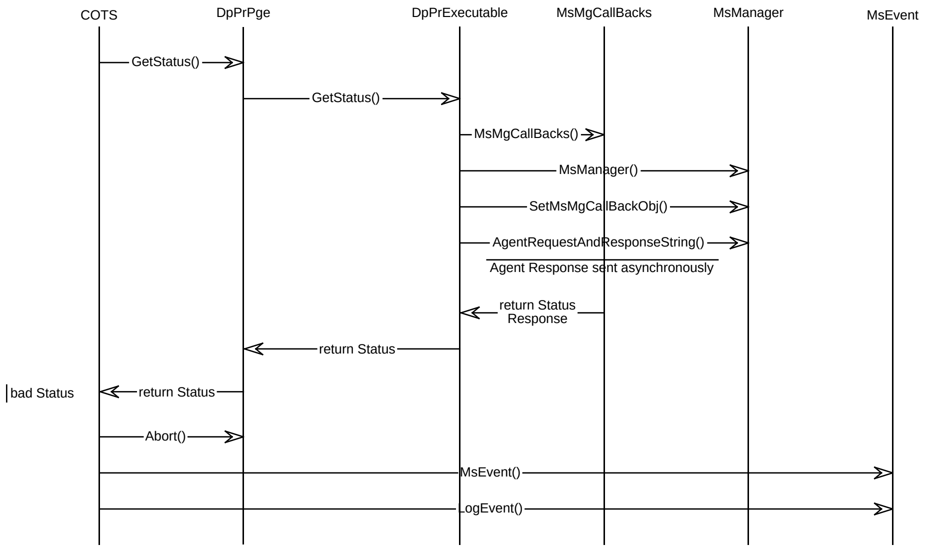


Figure 4.5-22. Case 2: Monitor Performance Trace

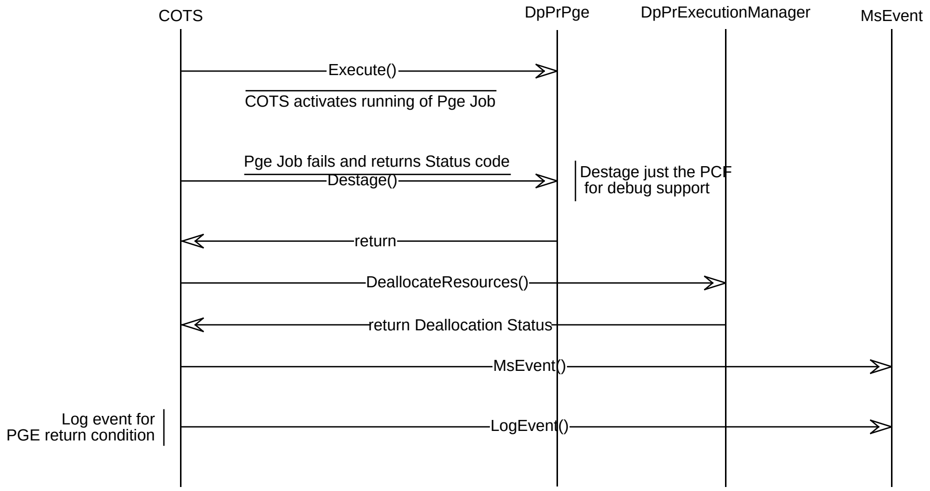


Figure 4.5-23. Case 3: Monitor Status Return Trace

4.5.4.3 Execution Post-Processing

4.5.4.3.1 Abstract

Execution Management provides services that are needed upon the completion of PGE execution. These services are used to initiate data destaging, creation of product history, deallocation of resources, etc. All generated outputs, Intermediate and Final Data Products, SDP Toolkit Process

Control and Status Message Log files, etc., will be destaged to the Data Server. The generated data products may be required as input to another PGE and therefore may not be removed from the local storage. The management of the data products will be necessary to insure that data staging of a data product is not performed excessively. Execution Management services will be required to generate processing-specific metadata. This metadata will be associated to a generated data product during the process of destaging to the Data Server. Any clean-up, such as deletion of input data and output data, may also be performed by the Processing CSCI during this phase.

4.5.4.3.2 Stimulus

The successful or unsuccessful completion of the execution of a PGE. The Execution Management services will perform these activities for each PGE which has completed execution.

4.5.4.3.3 Desired Response

All processing required at the completion of PGE execution will be performed. This includes initiation of data destaging, and other clean-up activities.

4.5.4.3.4 Participating Classes from the Object Model

- COTS
- DpPrPge
- DpPrExecutionManager
- DpPrResourcemanager
- DpPrDataManager

4.5.4.3.5 Scenario Description

a. Upon completion of PGE execution, the Execution Management Services will do the following:

1. Check the generated errors codes to determine if the PGE output is useful.
2. Generate processing-specific metadata for incorporation into a Production History File, to facilitate its association with the output products during the destaging operation. Production History File consists of information used to identify what occurred during execution. This includes actual resource usage, input data references, date and time of execution, and user define parameters. Most of the information located in the Process Control File and the Data Processing Request will be added to the Production History File.
3. If the PGE produced an output data product, i.e., Intermediate Data or Final Data Product, initiate destaging of the data products. In addition, destage other non-product data like Status Message Log files.

<u>NOTE:</u> The following two steps are performed internally by the COTS Scheduler as per the logic imparted during the job definition phase.

4. Determine if the PGE's generated output data is required by another scheduled PGE
5. Determine if the PGE's input data is required by another scheduled PGE

6. Destage the PGE's data mapping to preserve any runtime updates regarding Intermediate temporary files created during this run of the PGE, or requests for the transfer of runtime data files (this service, acquired through the SDP Toolkit, provides a mechanism for the preservation of critical data files used during the processing of the PGE).
7. If the input data is not needed to execute another PGE, deallocate the resources that were used for the input data.
8. The COTS Scheduler Log is updated to indicate the current status of the PGE. (i.e., Processing Status is "Complete").
9. Deallocate the processing resources used by the PGE.

b. End of "Execution Management (Execution Post-Processing).

4.5.4.3.6 Event Trace

Figure 4.5-24 shows the execution post processing event trace.

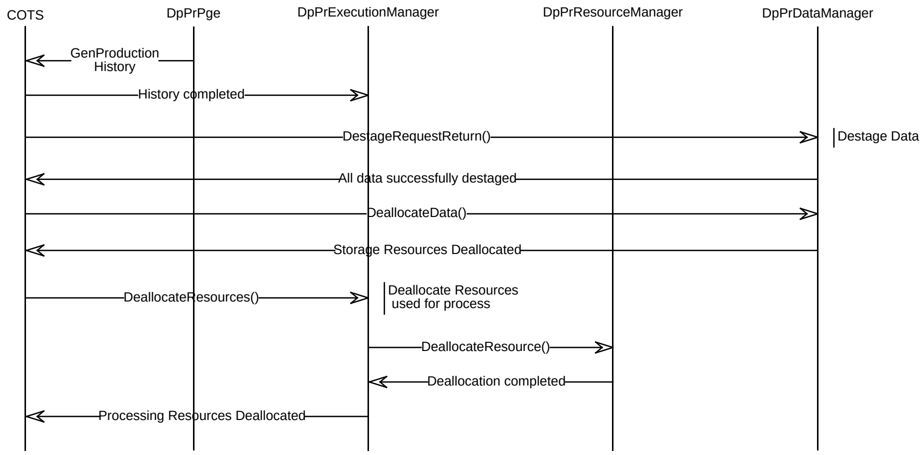


Figure 4.5-24. Execution Post Processing Event Trace

4.5.4.4 Failure of Execution

4.5.4.4.1 Abstract

The failure of the execution of a PGE may be caused by either an external fault, such as a resource failure, or by an internal fault due to a science algorithm error. In order to achieve the best response to a failed PGE, that PGE must terminate gracefully and return a status condition code which conveys the reason for the failure. The COTS Scheduler will determine, from this code, what further processing may be performed from the job interdependencies which are already defined, or through human interaction with the Alarm Manager utility. If a resource failed, there may be an attempt to move the PGE to similar resources in order to initiate execution of the PGE again. If the PGE failed because of an internal fault, an attempt will be made to determine the cause of this fault through the status condition code. If there is hope that the PGE may execute properly, the COTS Scheduler may initiate execution again.

4.5.4.4.2 Stimulus

The stimulus which causes a PGE to fail during execution comes from a fatal error condition that occurs during processing of the algorithm. Whether internal, or external to the PGE, the fault should be handled properly by the science software.

4.5.4.4.3 Desired Response

Upon detection of the fatal error condition, the error handling portion of the PGE should take-over to redirect processing, or exit gracefully; the latter case is depicted in this scenario.

4.5.4.4.4 Participating Classes from the Object Model

- COTS
- DpPrExecutable
- DpPrPcf
- DpPrPge
- DpPrExecutionManager
- DpPrResourceManager
- DpPrDataManager
- MsEvent
- PlDataGranule
- PlDpr
- DsClESDTReferenceCollector
- DsClRequest
- DsClCommand

4.5.4.4.5 Scenario Description

a. When the PGE fails due to a fatal error condition during processing, the following activities are performed:

    1. Retrieve the error condition code from the PGE.
    2. To signal that recovery attempts are underway, update the COTS Scheduler Interface to display "PGE Failed".
    3. Destage the Process Control File for this run to preserve a copy for SCF investigation.
    4. Destage whatever output data was produced. This data will not be archived necessarily, but may be used for investigation into the cause of the malfunction (the actual list of data to preserve is identified at runtime by the science software, through interaction with the SDP Toolkit; if nothing else, the PGE should handle the error to the point of being able to specify the set of data for post-mortem analysis).

5. At this point, the resources are still allocated and all of the input data is still staged. Just prior to resource deallocation, checks are performed on the health of the allocated resources to determine if the fault was external or internal to the PGE.
    (a) For the latter case, the COTS Scheduler proceeds with cleanup activities in order to make room for subsequent processing. If the cause was due to resource problems, the PGE may be re-initiated after sufficient resources are re-allocated.
    (b) If resources cannot be reallocated to re-initiate the PGE, then the job which represents the Data Processing Request for this PGE gets reprioritized within the COTS Scheduler, to be retried at a later time.
6. Deallocate all of the Resources used for this processing run.
7. Log a report on the health of currently allocated resources to MSS.
8. Also, send a message to the COTS Scheduler Log indicating that processing status is "Failure".

b. End of "Execution Management (Failure of Execution)."

4.5.4.4.6 Event Trace

Figure 4.5-25 shows the failure of execution event trace.

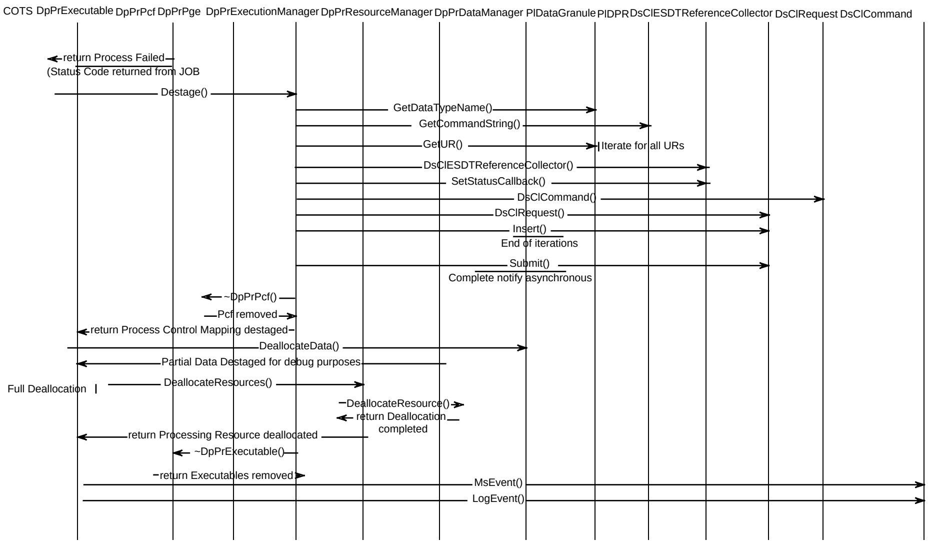


Figure 4.5-25. Failure of Execution Event Trace

4.5.4.5 Failure of Processing Resource

4.5.4.5.1 Abstract

If a resource fails during execution of a PGE, the COTS Scheduler will attempt to re-initiate the PGE using other suitable resources, if local policies so dictate. This will be done only if the resource information in the Data Processing Request is generic enough to allow this transferal. If the resource information is tied to a particular computer because of the unique characteristics of this computer, other error recovery actions, including Processing Operations staff intervention, may be initiated.

4.5.4.5.2 Stimulus

The stimulus which causes a PGE to fail during execution, due to a problem with a resource, originates from a fatal error condition that is detected during processing of the algorithm. Most likely, these type of external error conditions will not be overcome by the algorithm; the algorithm is still expected to fail in a graceful manner to ensure the return of an appropriate error condition code.

4.5.4.5.3 Desired Response

The execution of a PGE will be terminated by the error handling component of the PGE; all PGEs should contain a termination section built into the highest-level module of the PGE, through which control ultimately passes and a final process status gets defined (e.g., at the end of a PGE shell script). The resource which created the error condition should be replaced so that the PGE can be reactivated.

4.5.4.5.4 Participating Classes from the Object Model

- COTS
- DpPrExecutable
- DpPrPcf
- DpPrPge
- DpPrExecutionManager
- DpPrResourceManager
- DpPrDataManager
- MsMgCallBacks
- MsManager
- MsEvent

4.5.4.5.5 Scenario Description

a. When the PGE fails due to a resource-induced fatal error condition during processing, the following activities are performed:

   1. Send a PGE condition code "Failed due to resource" to the COTS Scheduler
   2. Signal that recovery efforts are under way - Update the COTS interface to display "PGE Failed"

3. Retrieve the error condition code from the PGE and activate the appropriate recovery job (this is performed automatically through predefined job definitions).
4. Destage whatever output data was produced. This data will not be archived necessarily, but may be used for investigation into the cause of the malfunction (the actual list of data to preserve is identified at runtime through science software interaction with the SDP Toolkit).
5. Destage the Process Control File (PCF) for this run to preserve a copy for SCF investigation.
6. At this point, the resources are still allocated and all of the input data is still staged. The Resource Management service is activated, as part of the data deallocation phase, to perform a check on the health of the allocated resources to determine if the fault was external or internal to the PGE.
    - (a) If the cause was due to resource problems, the PGE may be re-initiated after sufficient resources are re-allocated. For the latter case, the Resource Management service proceeds with cleanup activities in order to make room for subsequent processing.
    - (b) If resources cannot be re-allocated to re-initiate the PGE, the Data Processing Request for this PGE gets reprioritized within the COTS Scheduler to be retried at a later time.
7. Update the operational mode of the wayward resource to "OFFLINE."
8. Log a report on the health of currently allocated resources to MSS.
9. Deallocate the Processing resources. All CPU resources will be reclaimed but depending on the nature of the resource failure, only part of the PGE storage resources may be deallocated.
    - (a) If the cause of the resource failure was due to one particular disk device (i.e., file system) on the local host then simply deallocate that resource if it was allocated for the failed PGE. Only those PGE constituents which were resident on that resource will need to be restaged
    - (b) If however the entire host contributed to the resource problems, then all resources for that host will have to be deallocated. This means that all executables and Status Message Files for the PGE will have to be restaged to a different host for a subsequent run.
10. Send a message to the COTS Scheduler Log indicating that processing status is "Failure".

b. End of Scenario.

4.5.4.5.6 Event Trace

Figure 4.5-26 shows the failure of processing resource event trace.

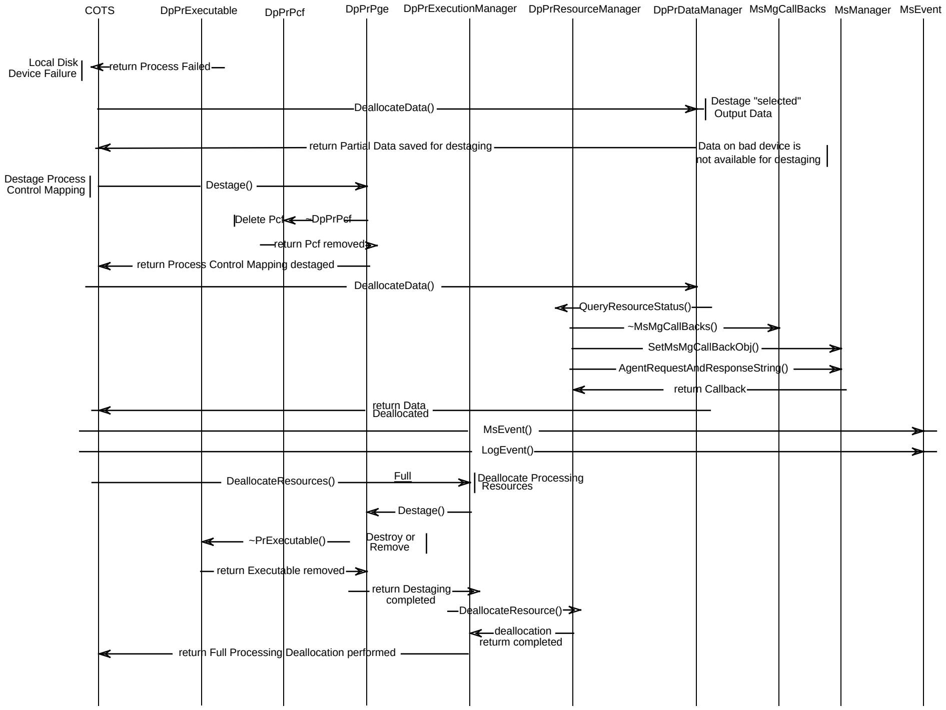


Figure 4.5-26. Failure of Processing Resource Event Trace

4.5.5 Resource Management Scenarios

Resource Management services are provided to manage and monitor the Data Processing subsystem resources which are used exclusively in the production environment. These services will be used to maintain the information needed to track the health and availability of these resources and coordinate their allocation and deallocation within the Data Processing subsystem.

Resource Management services include the following activities:

a. Initialization of Resource Management Information—This activity occurs at system initialization and on the update of resource availability information as it is received from MSS. This is the interface which will be used, in conjunction with MSS, to provide the initial resource configuration and updated resource information. See scenario "Resource Management: Initialization of Resource Management Information".

b. Modification of Resource Management Information—This activity occurs when resources are allocated or deallocated to support data staging/destaging and PGE execution. See scenario "Resource Management: Modification of Resource Management Information."

c. Query Resource Management Information—This activity occurs whenever Data Management is checking for available resources, or when processing resources are needed to stage a PGE and its components to a local resource. See scenario "Resource Management: Query Resource Management Information" for more details.

4.5.5.1 Resource Management Configuration Initialization

4.5.5.1.1 Abstract

The Planning and Data Processing resource characteristics are collected, maintained and used to determine whether or not sufficient resources are available to support the Data Processing subsystem production, such as the execution of PGEs and the staging of data for Data Processing Requests. The initialization of these resources is accomplished through interaction with MSS to create a working model of only those resources which may be used by Planning and Processing.

4.5.5.1.2 Stimulus

The stimulus for the initialization of the Resource Management Configuration is as follows:

The initialization of the Resource Management information is performed at Planning subsystem start-up through manual interaction with the Planning workbench and at production plan activation time.

4.5.5.1.3 Desired Response

A list of resources will be created and the data necessary to manage the resources, such as current state (OFFLINE, ONLINE, etc.) and characteristics (e.g., amount of usable disk space) will be recorded.

4.5.5.1.4 Participating Classes From the Object Model

- ResourceManager
- PlResourceUI
- DpPrResourceConfiguration
- MsDAAC
- DpPrString
- DpPrComputer
- DpPrDiskPartition
- DpPrDiskAllocation

4.5.5.1.5 Scenario Description

a. Through manual interaction with a Planning provided User Interface (UI), the Resource Manager activates the method to build the resource configuration.

b. Through the iterative application of a filtering method on MSS services, current DAAC resource information which applies to Planning and Processing can be retrieved.

    1. Create the appropriate resource objects using the configuration information obtained from MSS. Initialize data for each resource in the Data Processing subsystem.

        (a) Initialize Computer resource information—Retrieve machine type, operating system, number of CPUs, total RAM and current state.

(b) Initialize Disk Device resource information—Retrieve file system base paths, block size, total partition size and current state.

(c) Initialize Disk Allocation information—Retrieve all of the system uses for this Disk Device.

2. Create a String object and associate it with this Computer resource, or add the resource to the String if it's already defined.

b. End of Resource Management (Initialization of Resource Configuration Information).

4.5.5.1.6 Event Trace

Figure 4.5-27 shows the initialization of resource configuration information event trace.

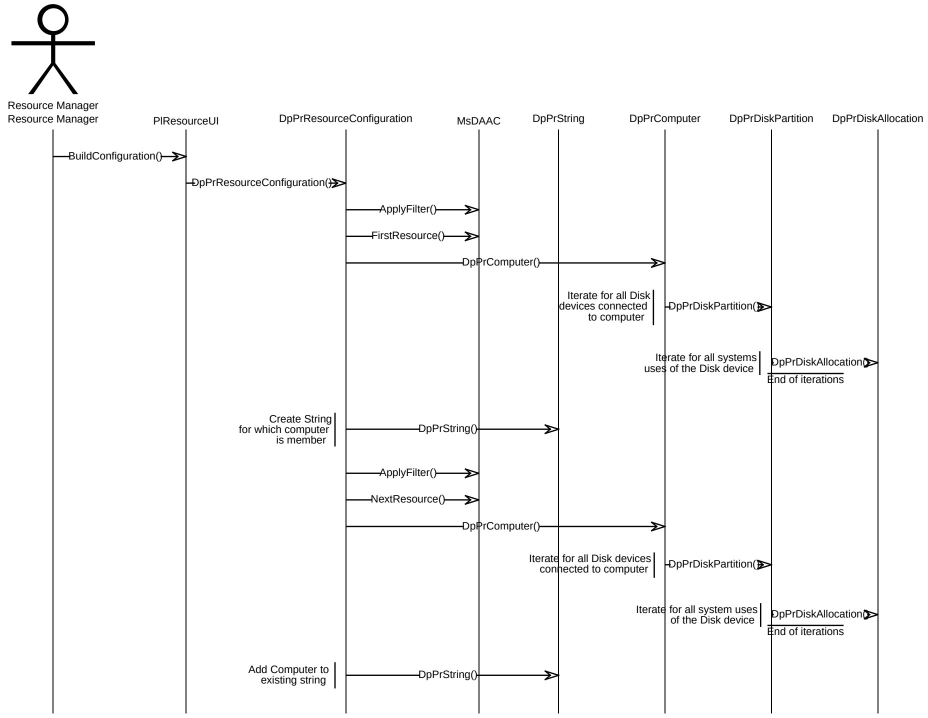


Figure 4.5-27. Initialization of Resource Configuration Information Event Trace

4.5.5.2 Modify the Resource Management Information

4.5.5.2.1 Abstract

The modification of the Resource Management information is performed whenever resources are used to support the Processing CSCI, such as at the beginning, or end of the execution of a COTS Scheduler defined job (i.e., Data Processing Request), or the beginning or end of the staging or destaging of data. The state of a resource may also change as a result of system error conditions, or as a resource is scheduled to be out of service.

4.5.5.2.2 Stimulus

The stimuli for the modification of the Resource Management List is the following:

a. Data Staging.

b. Data Destaging.

c. PGE Execution.

d. Resource Update Information obtained through MSS contact.

4.5.5.2.3 Desired Response

The entry of the resource to be modified will be updated to indicate current state of allocation, and/or changed resource characteristics.

4.5.5.2.4 Participating Classes From the Object Model

- COTS
- DpPrPge
- DpPrExecutionManager
- DpPrResourceManager
- DpPrDataManager
- MsMgCallBacks
- MsManager

4.5.5.2.5 Scenario Description

a. Whenever the COTS Scheduler executes a job to allocate or deallocate resources, or if an update to the status of resources has been acquired through MSS, do the following:

    1. If a Resource status update has been obtained from MSS then

        (a) Update the State of the Resource.

        Resource State = {ONLINE,OFFLINE, etc.}

    2. If Allocation of resources required for data staging and PGE Execution then

        (a) Update allocation information for the resources involved in the current processing.

            (1) Allocate disk space to support Data staging and Output Generation.

            (2) Allocate disk space required to support the hosting of the PGE's executables and runtime Status Message Files (SMFs). This step will only be necessary if the runtime files were not already resident due to first time run of the PGE, or prior resource problems which necessitated the removal of the files following a prior run.

(3) Allocate the disk space required to support the Process Control File (PCF) for this run of the PGE.

(4) Allocate the CPU(s) required to support PGE Execution.

(b) If Allocation failed due to resource problems then

(1) Obtain a resource status update from the MSS agent.

(2) Update the local resource status information and return an error condition to the COTS Scheduler.

(3) Depending on the error condition, re-allocation of resources may be performed to support the current processing (see 2a).

3. Following data destaging, if deallocation of resources required then

(a) Update information for the resources involved in the processing that just completed. If the same PGE will be executing again on the same host, disk space for the PGE's constituents (e.g., executable files) may not be deallocated.

(1) Deallocate disk space to support Data staging and Output Generation.

(2) Deallocate disk space required to support the Process Control File (PCF).

(3) Deallocate the CPU(s) required to support PGE Execution.

b. End of Resource Management: Modify Resource Management Information.

4.5.5.2.6 Event Trace

Figure 4.5-28 shows the modify resource information event trace.

4.5.5.3 Query the Resource Management List

4.5.5.3.1 Abstract

A query of the Resource Management information is performed to determine if resources are available to support the execution of a PGE. This scenario may also occur when checking to see if resource faults have occurred during PGE execution.

4.5.5.3.2 Stimulus

The stimuli for the query of the Resource Management information is the following:

This operation is performed whenever an allocation of processing resources is requested as a prelude to the execution of a PGE.

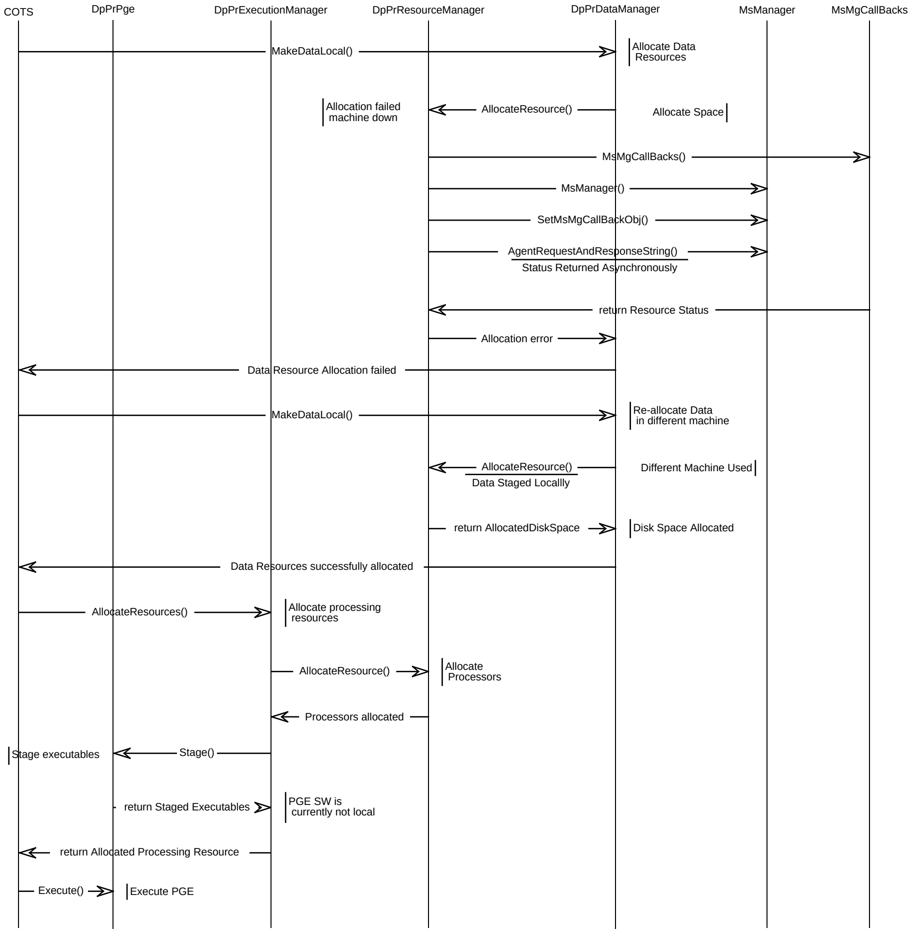


Figure 4.5-28. Modify Resource Information Event Trace

4.5.5.3.3 Desired Response

The information being requested for a set of resources will be produced.

4.5.5.3.4 Participating Classes From the Object Model

- COTS
- DpPrExecutionManager
- DpPrResourceManager
- DpPrComputer
- DpPrDiskPartition

4.5.5.3.5 Scenario Description

a. When the Execution Management services are performing the initiation of staging and execution of a PGE, determine whether the appropriate resources will be available to support PGE execution.

   1. Query for information on the set of resources required by the PGE.

      (a) Resource State {ONLINE, OFFLINE, etc.}

      (b) Number of available processors

      (c) Amount of available disk space

      (d) Memory configuration for the machine

   2. End of Resource Management (Query of Resource Management Information).

4.5.5.3.6 Event Trace

Figure 4.5-29 shows the query of resource management information event trace.

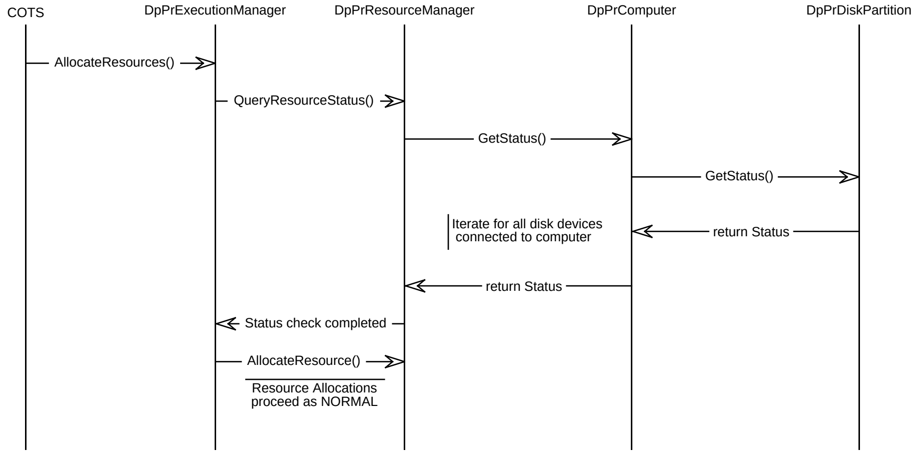


Figure 4.5-29. Query of Resource Management Information Event Trace

4.5.6 Quality Assurance Scenarios

The Quality Assurance Scenarios represent a high-level view of how DAAC manual quality assurance activities can be supported by the Processing CSCI. Since the process of DAAC manual quality assurance is still not a well understood process, the information presented in the scenarios will be limited, but will provide a starting point on what the Processing CSCI will support. The general approach to supporting DAAC manual quality assurance activities is to build on the support that SDPS is providing for SCF and user quality assurance activities. These activities are supported by providing data visualization tools, subscription submittal and withdrawal functions, and by allowing the updating of quality assurance metadata by authorized personnel. These activities are supported through Client provided functions. Any additional unique functions which are required only for DAAC manual quality will be provided by the Processing CSCI.

4.5.6.1 Q/A Subscription Submittal

4.5.6.1.1 Abstract

A Q/A position can subscribe to data products generated by the Processing CSCI's execution of a PGE. The Data Server then informs Q/A when the product has been generated and is available for review. This allows Q/A to be decoupled from the generation and initial storage of the data product. It also means that Q/A activities do not have to occur as soon as the product is generated.

4.5.6.1.2 Stimulus

Starting of the Q/A Monitor Command GUI with the intent of submitting a subscription will initiate the processing in this scenario.

4.5.6.1.3 Desired Response

The following actions will occur on the subscription request input:

a. The Data Server will be notified that Q/A wishes to subscribe to a data product.

b. The Data Server enters the subscription request.

c. The fact that the product is subscribed to is recorded in the PDPS database.

4.5.6.1.4 Participating Classes From the Object Model

a. DpPrQaMonitor

b. PlDataTypes

c. PlDataType

d. AdCollection

e. Advertisement

f. DsClSubscription

4.5.6.1.5 Scenario Description

a. The Q/A position brings up the Monitor Command Window and selects *Display Data Types for Subscription*.

b. The Data Type Selection Window comes up, displaying the data types which are not currently subscribed to.

c. The Q/A position selects the data type for subscription; the Data Type Selection Window goes away.

d. The Q/A position selects to *Submit the Subscription* from the Monitor Command Window.

e. A subscription (Data Server Client public class) is created with a command type of Submit and submitted to the Data Server.

f. The myQaSubscription flag in the Data Type information recorded in the PDPS database is set to *TRUE*.

4.5.6.1.6 Event Trace

Figure 4.5-30 shows the Q/A subscription submittal event trace.

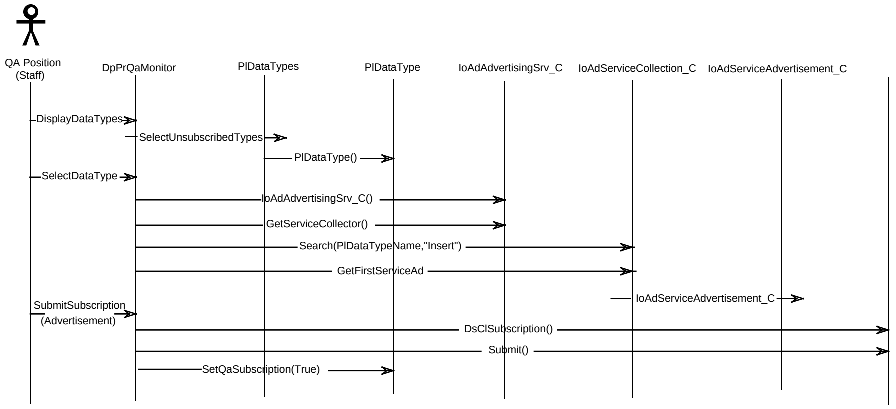


Figure 4.5-30. Q/A Subscription Submittal Event Trace

4.5.6.2 Q/A Subscription Withdrawal

4.5.6.2.1 Abstract

A Q/A position can withdraw subscriptions to data products generated by the Processing CSCI's execution of a PGE. The Data Server no longer informs Q/A when products of that data type have been generated.

4.5.6.2.2 Stimulus

Starting of the Q/A Monitor Command GUI with the intent of withdrawing a subscription will initiate the processing in this scenario.

4.5.6.2.3 Desired Response

The following actions will occur on the subscription withdrawal input:

a. The Data Server will be notified that Q/A wishes to withdraw its subscription to a data product.

b. The Data Server enters the subscription withdrawal request.

c. The fact that the product is no longer subscribed to is recorded in the PDPS database.

4.5.6.2.4 Participating Classes From the Object Model

a. DpPrQaMonitor

b. PlDataTypes

c. PlDataType

d. AdCollection

e. Advertisement

f. DsClSubscription

4.5.6.2.5 Scenario Description

a. The Q/A position brings up the Monitor Command Window and selects *Display Data Types for Subscription Withdrawal*.

b. The Data Type Selection Window comes up, displaying the currently subscribed-to data types.

c. The Q/A position selects the data type for withdrawal; the Data Type Selection Window goes away.

d. The Q/A position selects to *Withdraw the Subscription* from the Monitor Command Window.

e. A subscription (Data Server Client public class) is created with a command type of withdraw and submitted to the Data Server.

f. The myQaSubscription flag in the Data Type information recorded in the PDPS database is set to *FALSE*.

4.5.6.2.6 Event Trace

Figure 4.5-31 shows the Q/A subscription withdrawal event trace.

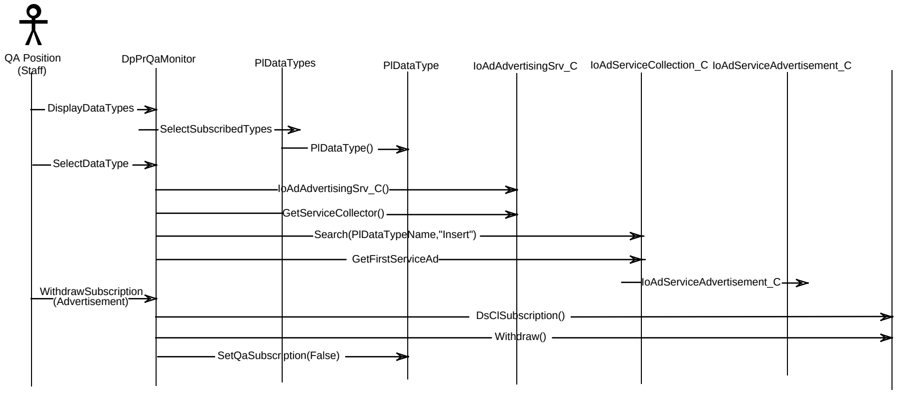


Figure 4.5-31. Q/A Subscription Withdrawal Event Trace

4.5.6.3 Q/A Get Data

4.5.6.3.1 Abstract

When a product has been generated by a PGE and is available for review, the Data Server checks to see if there are any Q/A subscriptions to that data. If there are, the Data Server sends e-mail to the Q/A position stating that the product has been generated and is available for review.

Then, when the Q/A position is ready to review the product, it starts up the Q/A Monitor Command GUI and can request the data from the Data Server.

4.5.6.3.2 Stimulus

Entering the GetData command in the Q/A Monitor Command GUI will initiate the processing in this scenario.

4.5.6.3.3 Desired Response

The following actions will occur upon the stimulus of this scenario:

a. Q/A enters a request for subscribed-to data.

b. The data is retrieved from the Data Server and returned to the Q/A position.

4.5.6.3.4 Participating Classes From the Object Model

a. DpPrQaMonitor

b. DsClESDTReference

c. PlDataGranules

d. PlDataGranule

4.5.6.3.5 Scenario Description

a. The Q/A position selects to *Retrieve Data* from the Monitor Command Window, using the UR obtained in the e-mail message.

b. The correct instance of the data type is retrieved from the Data Server.

4.5.6.3.6 Event Trace

Figure 4.5-32 shows the Q/A subscriptions event trace.

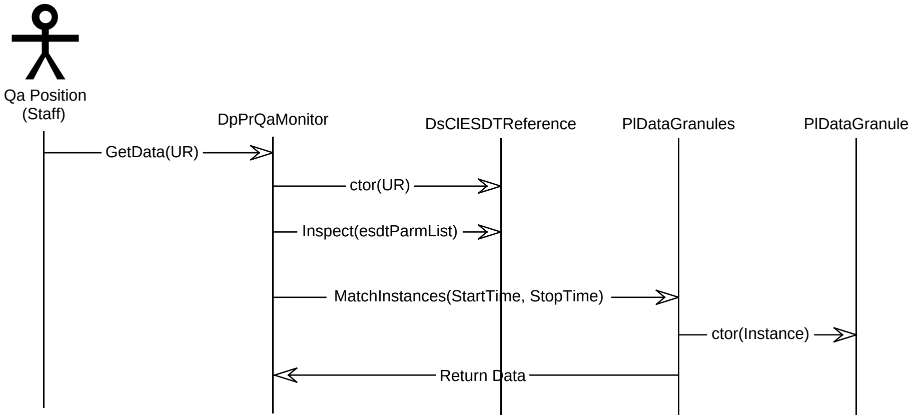


Figure 4.5-32. Q/A Subscriptions Event Trace

4.5.6.4 Q/A Visualize Data

4.5.6.4.1 Abstract

When a product has been generated by a PGE and is available for review, the Data Server checks to see if there are any Q/A subscriptions to that data. If there are, the Data Server sends e-mail to the Q/A position stating that the product has been generated and is available for review.

Then, when the Q/A position is ready to review the product, it starts up the Q/A Monitor Command GUI and can choose to visualize the data. This will retrieve the data from the Data Server and invoke EOSVIEW to display a visual interpretation of the product.

4.5.6.4.2 Stimulus

Entering the Visualize Data command in the Q/A Monitor Command GUI will initiate the processing in this scenario.

4.5.6.4.4 Desired Response

The following actions will occur upon the stimulus of this scenario:

a. Q/A enters a request to visualize subscribed-to data.

b. The data is retrieved from the Data Server and returned to the Q/A position.

c. EOSVIEW is invoked to display a visual rendition of the product.

4.5.6.4.4 Participating Classes From the Object Model

a. DpPrQaMonitor

b. DsClESDTReference

c. PlDataGranules

d. PlDataGranule

e. EOSVIEW

4.5.6.4.5 Scenario Description

a. The Q/A position selects to *Visualize Data* from the Monitor Command Window, using the UR obtained in the e-mail message.

b. The correct instance of the data type is retrieved from the Data Server.

c. EOSVIEW is invoked with the data retrieved from the Data Server.

4.5.6.4.6 Event Trace

Figure 4.5-33 shows the visualize science data event trace.

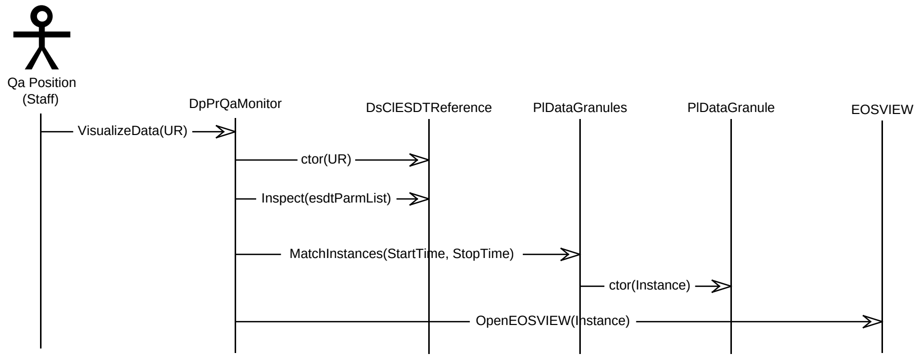


Figure 4.5-33. Visualize Science Data Event Trace

4.5.6.5 Q/A Metadata Update

4.5.6.5.1 Abstract

Q/A specifies metadata associated with data products to which it subscribes. This metadata describes format, amount, sampling information, etc. Q/A has the option of changing or updating the Q/A metadata associated with any data product.

4.5.6.5.2 Stimulus

Entering the Update Metadata command in the Q/A Monitor Command GUI will initiate the processing in this scenario.

4.5.6.5.3 Desired Response

The following actions will occur upon the stimulus of this scenario:

- The quality assurance metadata is updated, the metadata update will be stored with the product at the appropriate Data Server.

4.5.6.5.4 Participating Classes From the Object Model

a. DpPrQaMonitor

b. PlDataTypes

c. PlDataType

d. AdCollection

e. Advertisement

f. DsClESDTReferenceCollector

g. GlParameter

h. GlParameterList

i. DsClCommand

j. DsClRequest

4.5.6.5.5 Scenario Description

a. The Q/A position brings up the Monitor Command Window and selects *Display Data Types for MetaData Update*.

b. The Data Type Selection Window comes up, displaying the currently subscribed-to data types.

c. The Q/A position selects the data type for which the Metadata is to be updated; the Data Type Selection Window goes away.

d. The Q/A position selects to *Update Metadata* from the Monitor Command Window.

e. The Metadata Editor Window comes up, showing the current values for the fields which are valid for the user to change. The user can update any or all these fields, followed by selecting the *Update* function.

f. The Metadata Editor Window goes away.

g. The Q/A Monitor constructs and submits a request to the Data Server to update the existing Metadata for the product with the new Metadata.

4.5.6.5.6 Event Trace

Figure 4.5-34 shows the update Q/A metadata event trace.

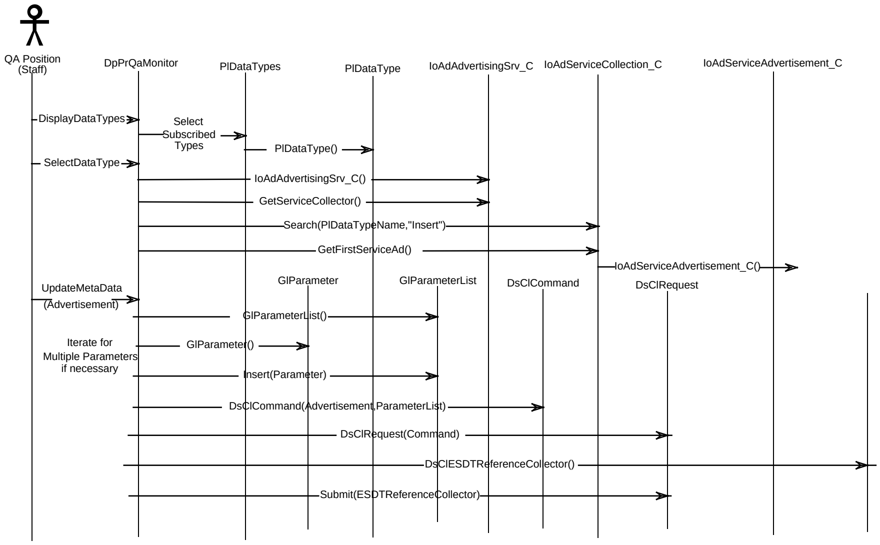


Figure 4.5-34. Update Q/A Metadata Event Trace

4.5.7 Data Pre-Processing Scenarios

The following scenario creates a preprocessed object by transforming a given data class. The scope of the scenarios includes all events impinging on or generated by certain objects in the Data Pre-Processing CSC. The software used to create pre-processed data to be used by a PGE should be thought of as a PGE. This PGE will be input into the Processing CSCI through the normal mechanisms provided to the Planning CSCI.

4.5.7.1 Scenario for Producing O/A Data Set and Level Zero Data Set

4.5.7.1.1 Abstract

Consider an example of CERES data preprocessing using SDPF-generated L0 data, and FDF-generated definitive orbit data. Level 0 CERES data received from SDPF will be received by the Ingest CSCI at the Goddard (GSFC) DAAC. The FDF-generated ephemeris data could arrive later than L0 data. Metadata are extracted for both ephemeris and L0 data and archived for later use. The attitude data along with position data are used to earth-locate each CERES footprint and calculate viewing geometry.

The ephemeris data from FDF will be in binary format as described in the FDF Generic Data Products Format ICD, 533-FDD-91/028, June 1991. There are two ways to handle the preprocessing of FDF-generated ephemeris file. Either it could be reformatted to HDF-EOS, appropriate metadata created and archived in the Data Server. If conversion to HDF is likely to degrade performance of the CERES algorithm, then it is stored in the Data Server in the original format. When a CERES Product Generation Executive (PGE) requests this ephemeris data, with information from the Planning subsystem, the data are retrieved from the Data Server, reformatted to the format of the hardware where the PGE will be executed. Appropriate metadata are prepared for the SDP Toolkit, and then staged for processing. This staged data is a new preprocessed object called "O/A Data Set".

When a CERES PGE requests L0 data, with information from the Planning subsystem, the Level 0 data and metadata are retrieved from the Data Server, and then staged for processing. This staged data is a new preprocessed object called "L0 Data Set".

4.5.7.1.2 Stimulus

Initiation of Pre-Processing Job by COTS.

4.5.7.1.3 Desired Response

Generation of pre-processed emphemeris and attitude data sets.

4.5.7.1.4 Participating Classes From the Object Model

4.5.7.1.5 Scenario Description

Scenario for Creating an Attitude File

a. Identify L0 Housekeeping data sets to process

b. Establish L0 Housekeeping data set processing order ("remove overlaps")

c. Find a boxcar averaging window number of attitude telemetry packets

d. Check the QAC list to determine quality flagging of telemetry packets

e. Initialize boxcar averaging of roll, pitch and yaw angles and their rates

    1. Check boxcar average to look for outliers in the angles and rates

    2. Write an attitude archive record

f. Find the next attitude telemetry packet

g. Check the QAC list to determine quality flagging of telemetry packets

h. Advance boxcar

    1. Check boxcar average to look for outliers in the angles and rates
    2. Write an attitude archive record

i. Exhaust attitude telemetry packets

j. Check the QAC list to determine quality flagging of telemetry packets

k. Terminate boxcar

    1. Check boxcar average to look for outliers in the angles and rates
    2. Write an attitude archive record

Scenario for Creating an Ephemeris File

a. Identify FDF data sets to process

b. Establish FDF data set processing order ("remove overlaps")

c. Read an FDF EPHEM format record

d. Unpack the FDF EPHEM format record orbital position and velocity vectors

e. Initialize boxcar averaging of the position and velocity vectors

    1. Check boxcar average to look for outliers in position and velocity vector
    2. Write ephemeris archive record

f. Advance boxcar

    1. Check boxcar average to look for outliers in position and velocity vector
    2 Write ephemeris archive record

g. Exhaust unpacked data points

h. Read next FDF EPHEM record

j. Unpack the FDF EPHEM format record orbital position and velocity vectors

k. Check for gaps in the position and velocity vector timelines

l. Advance boxcar

    1. Check boxcar average to look for outliers in position and velocity vector
    2. Write ephemeris archive record

m. Exhaust FDF EPHEM format records

n. Terminate boxcar

    1. Check boxcar average to look for outliers in position and velocity vector
    2. Write ephemeris archive record

4.5.7.1.6 Event Traces

Figure 4.5-35 shows the creating ephemeris file event trace. Figure 4.5-36 shows the creating attitude file event trace.

4.6 CSCI Structure

This section provides details on the underlying software structure of the Processing CSCI. The different components of the CSCI are defined, and a brief summary of the software architecture of the Planning and Data Processing Subsystems is provided.

The software architecture for the Planning and Data Processing Subsystems can be divided into three major layers of software:

a. PDPS User Interface Layer-contains the GUI applications used by the Operations staff to initiate Planning, Processing, and Algorithm Integration & Test activities.

b. PDPS Application Layer—contains the functional components for Planning, Processing, and Algorithm Integration & Test. These components work in collaboration with the GUI applications to perform Operations staff activities.

c. PDPS Persistent Data Layer—contains the persistent data structures required by Planning, Processing, and the Algorithm Integration & Test CSCIs. This data is retained within an SYBASE RDBMS. These data structures include the AutoSys Database schemas as well as the schemas required to support the Planning, Processing and Algorithm Integration & Test CSCI applications.

Figure 4.6-1 provides a diagram of the PDPS software architecture.

The Processing CSCI is decomposed into a number of CSCs. The CSCs correspond either to an application, or a class category describing a logically related set of functionality. The table below briefly outlines the CSCs defined for the Processing CSCI. Table 4.6-1 provides a brief description for each Processing CSC as well as a mapping to Custom or OTS software.

4.6.1 COTS CSC

4.6.1.1 Purpose and Description

The COTS CSC represents the COTS products, AutoSys and AutoXpert. A description of the components of AutoSys and the capabilities provided by AutoSys and AutoXpert are summarized in the following section.

There are three primary components of AutoSys:

a. AutoSys Database

b. Event Processor

c. Remote Agent

The AutoSys Database is the data repository for all system events as well as all job, monitor, and report definitions. This database is an RDBMS. For ECS, the RDBMS is SYBASE.

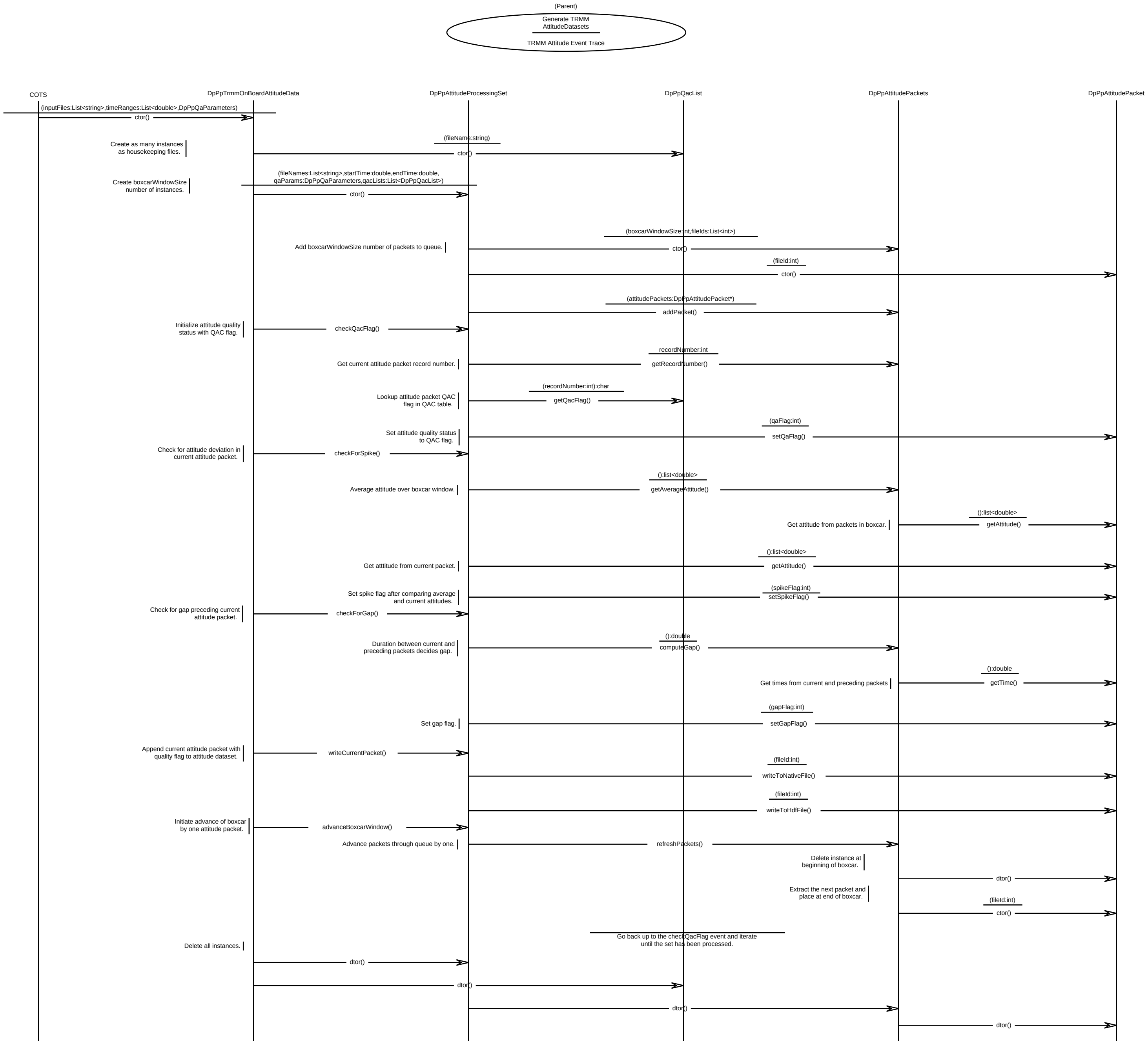

(Parent)
Generate TRMM AttitudeDatasets
TRMM Attitude Event Trace
COTS
DpPpTrmmOnBoardAttitudeData
DpPpAttitudeProcessingSet
DpPpQacList
DpPpAttitudePackets
DpPpAttitudePacket
(inputFiles:List<string>,timeRanges:List<double>,DpPpQaParameters)
ctor()
Create as many instances as housekeeping files.
(fileName:string)
ctor()
Create boxcarWindowSize number of instances.
(fileNames:List<string>,startTime:double,endTime:double, qaParams:DpPpQaParameters,qacLists:List<DpPpQacList>)
ctor()
(boxcarWindowSize:int,fileIds:List<int>)
Add boxcarWindowSize number of packets to queue.
ctor()
(fileId:int)
ctor()
(attitudePackets:DpPpAttitudePacket*)
addPacket()
Initialize attitude quality status with QAC flag.
checkQacFlag()
recordNumber:int
Get current attitude packet record number.
getRecordNumber()
(recordNumber:int):char
Lookup attitude packet QAC flag in QAC table.
getQacFlag()
(qaFlag:int)
Set attitude quality status to QAC flag.
setQaFlag()
Check for attitude deviation in current attitude packet.
checkForSpike()
():list<double>
Average attitude over boxcar window.
getAverageAttitude()
():list<double>
Get attitude from packets in boxcar.
getAttitude()
():list<double>
Get atttitude from current packet.
getAttitude()
(spikeFlag:int)
Set spike flag after comparing average and current attitudes.
setSpikeFlag()
Check for gap preceding current attitude packet.
checkForGap()
():double
Duration between current and preceding packets decides gap.
computeGap()
():double
Get times from current and preceding packets
getTime()
(gapFlag:int)
Set gap flag.
setGapFlag()
Append current attitude packet with quality flag to attitude dataset.
writeCurrentPacket()
(fileId:int)
writeToNativeFile()
(fileId:int)
writeToHdfFile()
Initiate advance of boxcar by one attitude packet.
advanceBoxcarWindow()
Advance packets through queue by one.
refreshPackets()
Delete instance at beginning of boxcar.
dtor()
Extract the next packet and place at end of boxcar.
(fileId:int)
ctor()
Go back up to the checkQacFlag event and iterate until the set has been processed.
Delete all instances.
dtor()
dtor()
dtor()
dtor()

Figure 4.5-35. Creating Ephemeris File Event Trace

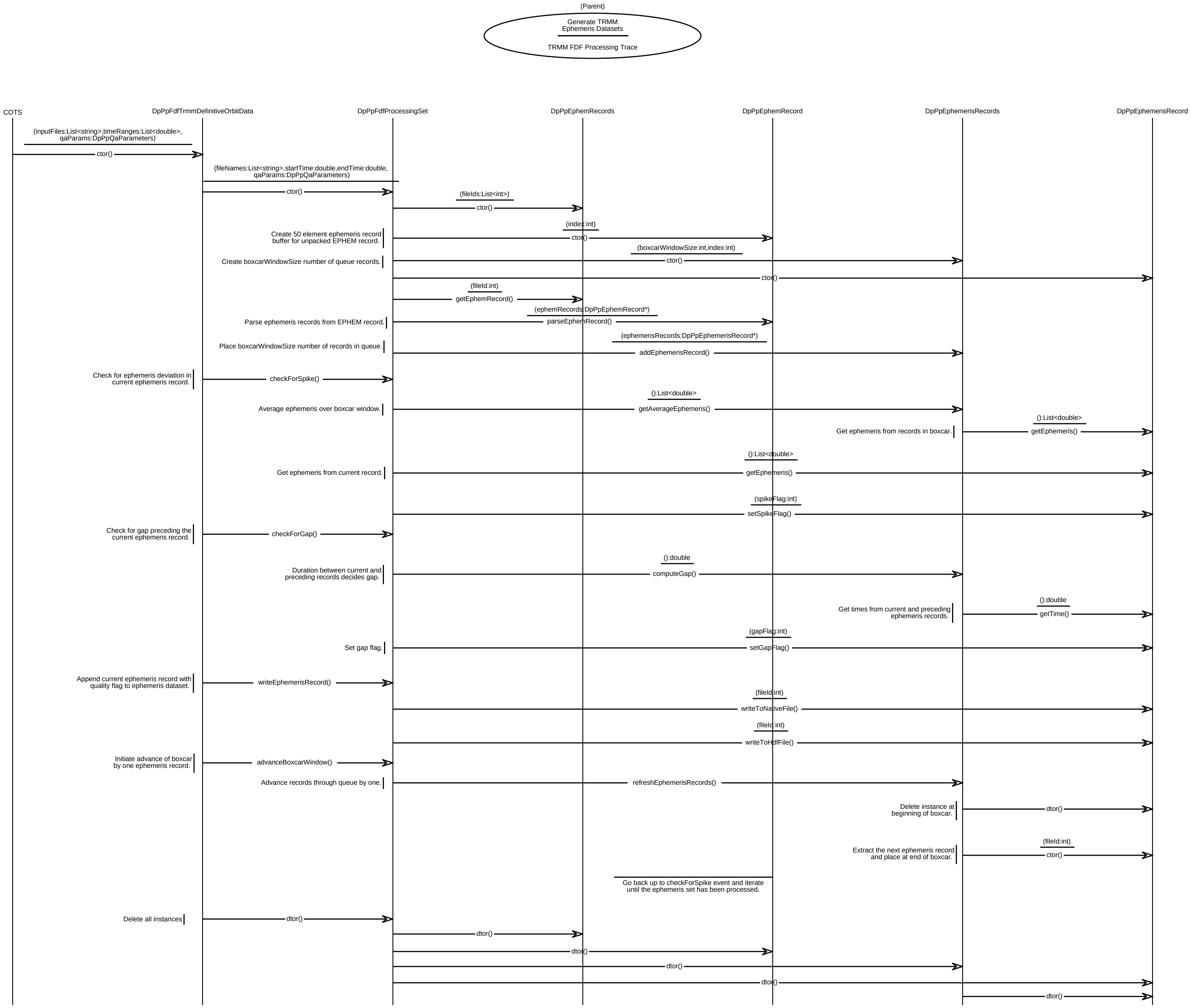

(Parent)
Generate TRMM Ephemeris Datasets
TRMM FDF Processing Trace
COTS
DpPpFdfTrmmDefinitiveOrbitData
DpPpFdfProcessingSet
DpPpEphemRecords
DpPpEphemRecord
DpPpEphemerisRecords
DpPpEphemerisRecord
(inputFiles:List<string>,timeRanges:List<double>, qaParams:DpPpQaParameters)
ctor()
(fileNames:List<string>,startTime:double,endTime:double, qaParams:DpPpQaParameters)
ctor()
(fileIds:List<int>)
ctor()
Create 50 element ephemeris record buffer for unpacked EPHEM record.
(index:int)
ctor()
Create boxcarWindowSize number of queue records.
(boxcarWindowSize:int,index:int)
ctor()
ctor()
(fileId:int)
getEphemRecord()
Parse ephemeris records from EPHEM record.
(ephemRecords:DpPpEphemRecord*)
parseEphemRecord()
Place boxcarWindowSize number of records in queue.
(ephemerisRecords:DpPpEphemerisRecord*)
addEphemerisRecord()
Check for ephemeris deviation in current ephemeris record.
checkForSpike()
Average ephemeris over boxcar window.
():List<double>
getAverageEphemeris()
Get ephemeris from records in boxcar.
():List<double>
getEphemeris()
Get ephemeris from current record.
():List<double>
getEphemeris()
(spikeFlag:int)
setSpikeFlag()
Check for gap preceding the current ephemeris record.
checkForGap()
Duration between current and preceding records decides gap.
():double
computeGap()
Get times from current and preceding ephemeris records.
():double
getTime()
Set gap flag.
(gapFlag:int)
setGapFlag()
Append current ephemeris record with quality flag to ephemeris dataset.
writeEphemerisRecord()
(fileId:int)
writeToNativeFile()
(fileId:int)
writeToHdfFile()
Initiate advance of boxcar by one ephemeris record.
advanceBoxcarWindow()
Advance records through queue by one.
refreshEphemerisRecords()
Delete instance at beginning of boxcar.
dtor()
Extract the next ephemeris record and place at end of boxcar.
(fileId:int)
ctor()
Go back up to checkForSpike event and iterate until the ephemeris set has been processed.
Delete all instances
dtor()
dtor()
dtor()
dtor()
dtor()
dtor()

Figure 4.5-36. Creating Attitude File Event Trace

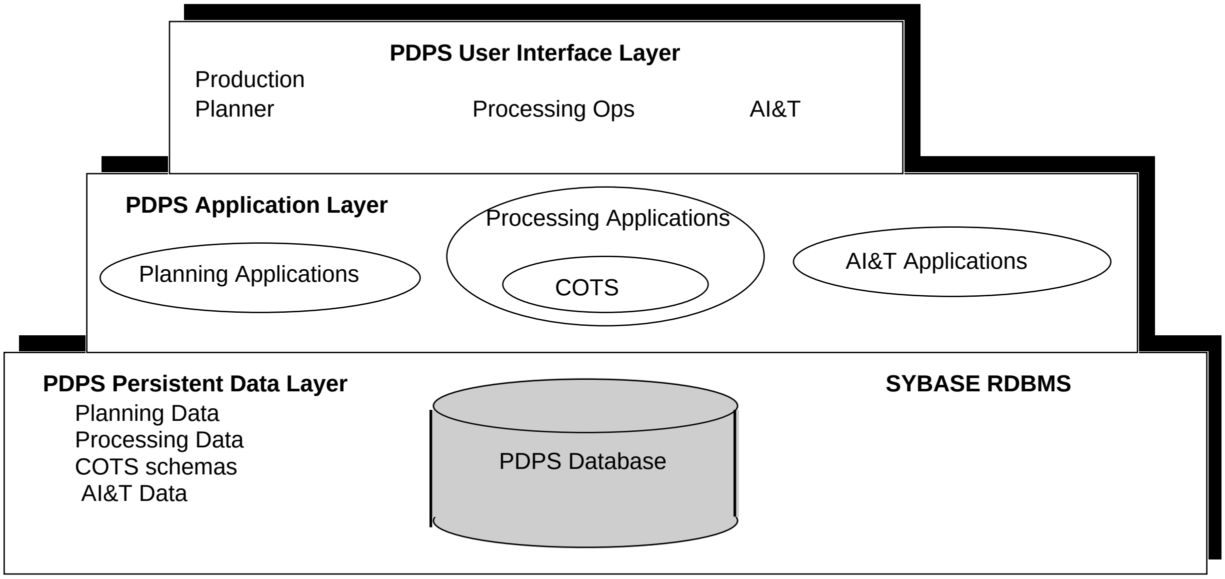


Figure 4.6-1. PDPS Software Architecture

Table 4.6-1. Processing CSCI Components (1 of 2)

CSC	Description	Type (Custom=DEV; off-the-shelf=OTS)
COTS	This CSC is used to represent the COTS products, AutoSys and AutoXpert. AutoSys has three major components; AutoSys Database, Event Processor, and the AutoSys Remote Agent. AutoSys is responsible for providing job management functions for the Processing CSCI. Also, AutoSys provides two GUIs; the Operator Console and AutoXpert. See section 4.1.3 for more information on AutoSys and AutoXpert capabilities.	OTS
COTS Management	Provides services required to interface with the COTS product, AutoSys. All unique AutoSys command-line interfaces and APIs are encapsulated into this collections of classes.	DEV
Resource Management	Provides services for the management of Science Processing Hardware resources. A low-level component used by the PGE Execution Management and Data Management CSCs to allocate, deallocate, and monitor the using of Science Processing Hardware resources.	DEV
Data Management	Provides services to manage the data required to support the execution of a PGE.	DEV

Table 4.6-1. Processing CSCI Components (2 of 2)

CSC	Description	Type (Custom=DEV; off-the-shelf=OTS)
PGE Execution Management	Provides services required to prepare a PGE for execution and perform post-processing activities.	DEV
Data Pre-Processing	Provides data pre-processing services to prepare ephemeris, O/A, and other ancillary data to be used by the PGE.	DEV
Quality Assurance Monitor Interface	The HMI used for performing DAAC manual quality assurance activities.	DEV

The Event Processor is the heart of AutoSys; it interprets and processes all the events it reads from the AutoSys Database. Sometimes called the event-daemon, the Event Processor is the program, running as a UNIX process, which actually runs AutoSys—it schedules and starts jobs. When started, the Event Processor continually scans the database for events to be processed. When it finds one, it checks whether the event satisfies the starting conditions for any job in the database. Based on this information, the Event Processor first determines what actions are to be taken, then instructs the appropriate process (or Remote Agent) to perform the actions. These actions might be the starting or stopping of jobs, checking for resources, monitoring existing jobs, or initiating corrective procedures.

The Remote Agent is a temporary process started by the Event Processor in order to perform a specific task on a remote machine. It starts the UNIX command specified for a given job, sends information about the task as event to the database, then exits. If the Remote Agent is unable to transfer the information, it waits and then tries again.

To support fault tolerance, AutoSys provides a high availability option which consists of a shadow database server plus a shadow event processor. In the event of a failure of the primary AutoSys Database or Event Processor, the shadow AutoSys Database or Event processor will take over its assigned role.

AutoSys is completely event-driven; i.e., to become activated by the Event Processor, an event must occur for a job. For example, the starting data and time have arrived, a prerequisite job has been completed, or a prerequisite file has been received. These events may come from a number of sources, such as:

a. Jobs changing states, such as starting, finishing successfully, finishing unsuccessfully, etc.

b. Data and Time.

c. Internal AutoSys verification agents, such as detected errors.

d Events sent with the **sendevent** command (AutoSys API) - either from the command line, or from user applications.

The following diagram, pictured in Figure 4-6.2 provides a sample scenario which describes the interactions between the three primary components of AutoSys; the AutoSys Database, the Event Processor, and the Remote Agent. This explains the steps taken to initiate a job which is defined in AutoSys. The scenario steps are the following:

a. From the RDBMS, the Event Processor reads a new event - a "start job" whose start time has arrived. It reads the appropriate job definition from the database, and based on that definition, determines what action to take (e.g. run the associated command line on Science Processor).

b. The Event Processor communicates with the Remote Agent on the Science Processor. As the Remote Agent receives the instructions from the Event Processor, the connection between the two processes is dropped. Therefore, even if the Event Processor disappears the remote machine is unaffected.

c. The Remote Agent performs such resource checks, such as ensuring that the minimum specified number of processes are available, then "forks" a child process, which will actually run the specified command.

d. The UNIX command completes and exits. The Remote Agent captures this exit code.

e. The Remote Agent communicates the event (exit code, status, etc.) directly to the RDBMS. If the RDBMS is unavailable for any reason, the Remote Agent will go into a wait/resend cycle until the message is delivered.

Only two AutoSys processes need to be running; i.e., the Event Processor and the RDBMS. When these two components are running, AutoSys is fully operational. The Remote Agent is started on a remote machine on an "as needed" basis. As soon as it completes its assigned task, it exits. Please note that the Remote Agent gets started on the remote machine by the Event Processor talking to the internet daemon (INETD) on the remote machine.

For details on the capabilities on AutoXpert, please see Section 4.1.4.3, Platinum Technology's AutoXpert. In terms of the PDPS Preliminary Design Specification, AutoXpert provides functions which were previously mapped to the Production Management CSC in the Planning CSCI. Since these functions are being provided by the AutoSys and AutoXpert products, it seemed desirable to map the COTS product to one CSCI. This decision eliminated the need to represent interfaces between AutoSys and AutoXpert across CSCI and Subsystem boundaries.

4.6.2 COTS Management CSC

4.6.2.1 Purpose and Description

The COTS Manager is a collection of classes used to interface with AutoSys. This is a part of the Production Planning Workbench application. This collection of classes encapsulates the unique AutoSys command-line interfaces and APIs and provides operations required by Planning to create, modify, cancel, release, and request status of jobs which exist in the AutoSys Database. Operations are also provided to notify AutoSys of an external events such as resource fault information.

4.6.2.2 Object Model Mapping

Please refer to Section 4.3.3 for the Cots Management CSC Object Model.

4.6.2.3 Candidate Products

There are no candidate products for this CSC. This is a custom component required to interface with ECS specific services.

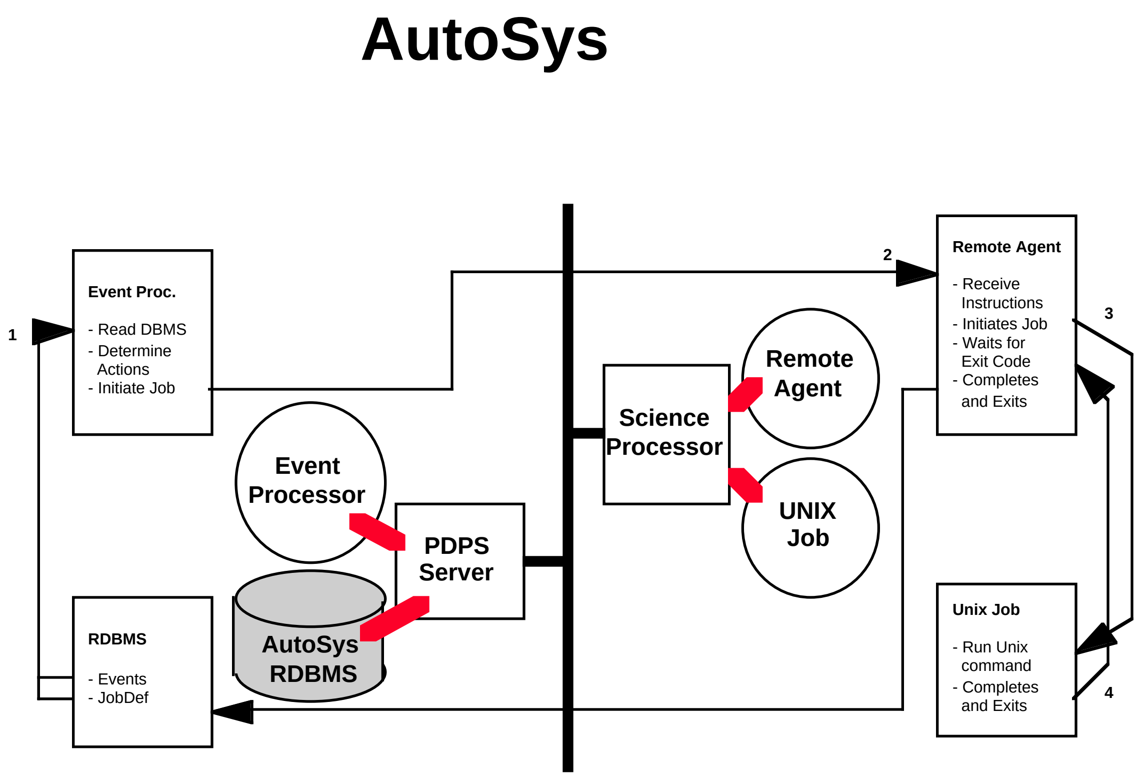


Figure 4.6-2 Interaction of AutoSys' Database, Event Processor, and Remote Agent

4.6.3 Resource Management CSC

4.6.3.1 Purpose and Description

The Resource Management CSC provides the services required for managing the Science Processing Hardware resources used for the generation of data products. This CSC manages the storage devices and computers which comprise the Science Processing Hardware resources. These Services support the allocation and deallocation of resources required for the execution of a PGE. Also provided are operations to initialize a logical mapping of the Science Processing Hardware resources. Currently, AutoSys provides some resource management capabilities. These services will augment the capabilities of AutoSys.

4.6.3.2 Object Model Mapping

Please refer to Section 4.3.6 for the Resource Management CSC Object Model.

4.6.3.3 Candidate Products

As stated before, AutoSys provides some limited resource management capabilities. ECS is committed to working with the vendor to influence the future direction of their product to meet the needs of the ECS science processing environment management.

4.6.4 Data Management CSC

4.6.4.1 Purpose and Description

This application manages the staging, destaging, and retention of data on Science Processing Hardware resources. The interface to the Science Data Server CSCI to stage (acquire) and destage (insert) data is provided by this application. This is a distinct application which is initiated as precursor job for a PGE. This job would be initiated by AutoSys when the dependencies defined for this job have been met. This application interfaces with the PDPS Database to retain persistent data on what data currently resides on Science Processing Hardware Resources.

4.6.4.2 Object Model Mapping

Please refer to Section 4.3.4 for the Data Management CSC Object Model.

4.6.4.3 Candidate Products

There are no candidate products for this CSC. This is a custom component required to interface with ECS specific services.

4.6.5 PGE Execution Management CSC

4.6.5.1 Purpose and Description

This application performs preparation and post-processing activities to support the execution of the PGE. These activities include the following:

a. Staging the executables and binaries which define the PGE, if required.

b. Creating the Process Control File (PCF) used to support the PGE. PCF contains input and output data information.

c. Preparing the Production History File to be destaged (inserted) into the Science Data Service CSCI with the science data products. The Production History File will contain information used as inputs to the PGE (the Data Processing Request information) as well as resource utilization information.

d. Deallocation of resources at the completion of data destaging.

4.6.5.2 Object Mapping

Please refer to Section 4.3.5 for the Data Management CSC Object Model.

4.6.5.3 Candidate Products

There are no candidate products for this CSC. This is a custom component required to interface with ECS specific services.

4.6.6 Data Pre-Processing CSC

4.6.6.1 Purpose and Description

The Data Pre-Processing CSC provides a set of services which allow the use of Orbit and Attitude (O&A) data, Level O, and Ancillary data products as inputs into the generation of EOSDIS data products. The services provided will modify the format of the data for which the PGE is expecting.

Data Preprocessing can be defined as preliminary processing or application of operations on a data set which do not alter or modify its scientific content. Preprocessing includes changes to the format of a data set by reordering the lower level byte structure, reorganization of a data set (ordering data items within and between physical files), preparing additional metadata based on lower level metadata, etc. There is a clear need for preprocessing. Data coming from various sources will be in numerous formats, containing a variety of metadata information that may or may not be suitable during certain stages of data handling. Due to a large number of potentially heterogeneous data sets, it is unlikely that they will be acceptable to services within ECS in their original form. As data move within the system, these incompatibilities can create obstacles to the smooth and efficient processing of data. The Preprocessing functions act on these data, and by introducing consistency make product generation seamless. It is important to distinguish Preprocessing as a logical group of processing functions and the Preprocessing CSC that contains some of these functions.

The data that needs preprocessing are:

- External ancillary data that are required as input for the generation of Standard Products. These data are non-EOS data (e.g., data from the National Oceanic and Atmospheric Administration (NOAA)).
- Attitude data contained in the spacecraft ancillary packet within Level Zero (L0) data from the Sensor Data Processing Facility (SDPF) for Tropical Rainfall Measuring Mission (TRMM).
- Orbit/Attitude (O/A) data contained in the spacecraft ancillary packet within L0 data from EOS Data and Operations Systems (EDOS) for EOS-AM.
- L0 Data Header received from EDOS for EOS-AM.
- Repaired orbit data generated by the Flight Dynamics Facility (FDF) for EOS-AM as replacement for defective onboard orbit data. For TRMM, the definitive orbit is FDF-generated but comes via SDPF.

The major functions that will be performed by Data Pre-Processing CSC are:

- Reformat FDF ephemeris data sets to Hierarchical Data Format with EOS extensions (HDF-EOS) format.
- Prepare additional metadata required by the Science Data Processing (SDP) Toolkit. The Preprocessing functions will derive any additional metadata from existing metadata to provide to the SDP Toolkit.
- Provide the SDP Toolkit L0 header and O/A data in a compatible format.
- Extract additional metadata (in addition to metadata extraction at ingest) to support services on certain ancillary data sets as necessary.

4.6.6.2 Object Model Mapping

Please refer to section 4.3.8 for the Data Pre-Processing CSC Object Model.

4.6.6.3 Candidate Products

There are no candidate products for this CSC. This is custom software.

4.6.7 Quality Assurance Monitor CSC

4.6.7.1 Purpose and Description

The Quality Assurance Monitor CSC is defined as the services required for DAAC manual quality assurance activities. The Quality Assurance Monitor software provides the Human-Machine Interface (HMI) and other support activities necessary to support DAAC manual quality assurance. This application is a completely separate from AutoSys. These DAAC manual quality assurance activities occur in a non real-time environment, i.e., there are no inherent dependencies between the performance of DAAC manual quality assurance activities and the active daily job schedule. Any quality assurance dependencies are handled through the Planning CSCI subscription Manager component. This application requires capabilities to subscribe to data and to be notified of the availability of data which exists at a Science Data Server CSCI. Also, this application uses public classes provided by the Science Data Server CSCI to update quality assurance metadata for a given science data product.

4.6.7.2 Object Model Mapping

Please refer to Section 4.3.7 for the Data Pre-Processing CSC Object Model.

4.6.7.3 Candidate Products

There are no candidate products for this CSC. This is a custom component required to interface with ECS specific services.

4.7 Processing CI Management and Operations

The following sections discuss the management and operation of the Processing CSCI, addressing how the CSCI is managed at the local and system levels. The Processing CSCI in relation to the system management strategy is discussed in Section 4.7.1. The approach to the development of operator interfaces for the Processing CSCI is then described in Section 4.7.2. Finally, the approach to reporting for the Processing CSCI is described in Section 4.7.3. This section addresses the operations and management activities as they relate to the Processing CSCI of the Processing Subsystem. The other major software component of the Subsystem, the AITTL CSCI, is discussed in Section 7.6.

The role of Production Management is discussed by providing information on the following topics:

- Operations activity level
- Interaction with Planning CSCI
- Use of AutoSys and AutoXpert for queue management and monitoring
- Resource Management

- QA Monitoring

Use of AutoSys and AutoXpert

A significant management concept for the Processing CSCI is the use of the AutoSys and AutoXpert job scheduling and monitoring software packages as a central component of the subsystem. These tools provide significant capabilities for the scheduling, execution, and monitoring of the production processing workload. The tools provide a reliable, off-the-shelf mechanism to define the processing tasks in an extremely flexible fashion, meeting all of the needs of the science software for controlling complex processing flows from one executable to another within a PGE. Capabilities such as AutoSys' 'Job Boxes' provide the needed mechanisms for job dependency definition. This capability to address complex job dependencies insures that DAAC management has the ability, now and for future instruments, to support the data processing schemes of science software developers. Additionally, these tools provide for job error handling, logging of job status and some resource management capabilities, which are extended as required by ECS custom software.

The operator interface provided by AutoXpert provides three views of the processing activities: the Timeline view depicting the development of processing over time, the Job Network view depicting the relationships between jobs and job boxes, and the Resource view which depicts the allocation of processing activities to processors within the system. Note that the AutoXpert provides the capability to perform limited 'what-if' capabilities to assess the impact of delayed processing on downstream activities. In addition, within AutoSys, the operators are able to modify job characteristics of individual jobs, such as the modification of priorities associated with a job. These tools provide significant capabilities for operations to manage and control processing activities within the ECS.

Operations Activity Level

A key design consideration for the Processing CSCI is that the operations personnel will interact with individual science data processing jobs on an exception basis only. No interactions is required of the operations personnel for normal operations, except for those required by the science software itself. No operations actions are required to start or stop a job, or to stage or destage data files. Operations personnel are provided with job monitoring tools through the AutoXpert system allowing them to observe the progress of the job through the system. Operations personnel are expected to monitor processing for anomalies that may require their attention. The AutoXpert tool provides alerts to operations for predefined conditions, such as processing jobs that have run beyond the expected end time.

Interaction with the Planning CSCI

The Processing CSCI is matched with the Planning CSCI to manage and control the significant science data processing activities allocated to the DAACs. The preparation or definition of processing activities to be accomplished is performed by the Planning CSCI. When the planning for the processing tasks has been completed and the necessary data has arrived to initiate the processing, the Planning CSCI releases the processing task to the Processing CSCI for subsequent management through the processing activities. This cooperative management of the planning and execution activities is a key feature of the management of science data processing.

Resource Management

The Processing CSCI provides resource management capabilities to control the allocation of processing tasks to processors. This capability allocates and deallocates processing resources as required for PGE Execution Management and Data Management CSCs. This capability maintains information on the processors allocated to the Processing CSCI for use in the management of science data processing. As tasks terminate and processors become available for other activities, the resource management function keeps track of the state of the processors. The Processing CSCI monitors the use of resources by the PGEs during their execution. This information is used to project the performance of the PGEs for the purpose of planning future processing.

QA Monitoring

The Processing CSCI includes tools to support quality assurance monitoring of the products generated during science processing. These capabilities include subscription services to access the data files that are to be quality checked, data visualization tools, and interfaces to update metadata for the products with QA information. This collection of utilities are a part of the essential items needed to manage the data quality checking operations that are an aspect of the end-to-end science data processing activity.

4.7.1 Processing CSCI and the System Management Strategy

The system management strategy as supported by the Processing CSCI is discussed in the following paragraphs.

4.7.1.1 System Management and Operations Philosophy

A primary design consideration for the Processing CSCI from the point of view of system management and operations is that operator interactions with Processing be simplified and automated as much as possible. In addition, the Processing CSCI provides operations with the needed flexibility to respond to unexpected tasks as they arise. The objectives of simplification and flexibility are attained, in large part, through the use of the AutoSys/AutoXpert scheduling tool.

4.7.1.2 Processing CSCI and the System

The following paragraphs discuss key features of the Processing CSCI in relation to system management.

Processors and Job Allocation

A key design feature of the Processing Subsystem is that only a single science processing job or PGE is allocated to a processor at a time. Because most processing systems are operated in a multiprocessing mode (i.e., multiple jobs simultaneously submitted for processing with swapping between jobs), the following paragraphs how this decision was reached.

Of the more than 120 PGEs evaluated (through AHWGP provided materials) for Release A and B, four are I/O-limited and the remainder are compute-limited. In most cases, then, there is no advantage to dispatching more than one PGE on the same processor at the same time. The multi-processing scheduling algorithm employed by most operating systems is sensitive to the I/Os that a processing task performs - when a task that has access to the processor attempts to perform a disk I/O, the task will be swapped out while the I/O completes and another task waiting for the processor is swapped in. Other factors are reflected in task scheduling but this is a key factor. This scheme works well as long as a mixture of I/O-bound and CPU-bound jobs are submitted together for execution—the jobs can overlap I/Os with CPU activity thereby resulting in a net speedup in processing. For the job mix anticipated for the ECS, however, the unbalanced mix does not provide the overlapped I/O and CPU jobs as most tasks would be waiting for the processor.

However, information is maintained for each PGE describing the PGE characteristics. These characteristics include information describing the degree to which the PGE is I/O-limited or CPU-limited. Because the information is available describing PGE characteristics, operations may modify the process scheduling control parameters to allow a collection of jobs with a mixture of CPU-limited and I/O-limited jobs to be allocated to a single processor. The decision to make this change in job allocation strategy can be made during operations based on actual experience with the PGE execution characteristics.

Resource Management and MSS

The Processing CSCI includes capabilities to control the processing resources available to it for science processing. Configuration information received from MSS is used to define the resources available. The current schedule of ground events (e.g., hardware maintenance activities) is also used as a part of the job scheduling process by reserving particular processors for ground events. The Processing CSCI then participates with the MSS in the management of the system processing resources to provide significant flexibility for resource management.

Data Staging and Destaging

The Processing CSCI, in conjunction with the Planning CSCI, provides for management of science data files between related processing jobs. In the general case, a science processing PGE requires several data files, which must be staged from the Data Server/Ingest prior to job execution. However, most processing jobs require data files that are the product of another PGE. For example, a Level 1a processing job might produce a Level 1a data file that is used as input to a Level 2 processing job. The Processing CSCI attempts to schedule processing jobs so that data files do not need to stage input files from the Data Server if the files have recently been produced and reside on processor local disks. This eliminates needless delays for data file staging. By providing this broader view of data file management, the Processing CSCI contributes to improved systems management.

Enterprise Management and Processing

As with the other components of the ECS, the Processing CSCI contributes to the enterprise management approach to system management. The Processing CSCI interacts with the MSS for startup and shutdown procedures. The Processing CSCI uses MSS provided services for the detection and management of faults. It also cooperates with MSS for other enterprise capabilities including logging of events and providing system management information, such as accounting information. The Processing CSCI also receives information from the MSS for resource management, as discussed earlier. In this way the Processing CSCI participates with the other ECS CSCIs to provide a systematic approach to system management.

PGE Interactions

To include the PGEs into the system management approach, the Processing CSCI provides access to the PGE for management of the PGE execution. The PGE Process Control File that is used by the PGE to receive control information is constructed by the Processing CSCI based upon directions received from the science software developers at the AI&T event. The Processing CSCI also provides an interface to the Data Server staging the PGE if necessary. This scheme incorporates the PGE into the system management scheme.

4.7.2 Operator Interfaces

This subsection describes the operator user interfaces provided by the Processing CSCI to operations personnel. A general description of the framework and methodology employed for the development of these interfaces can be found in Section 4.7. of the Detailed Design Overview (305-CD-001-001). This subsection augment that information with additional design information which is specific to the Processing CSCI.

The operator user interfaces for the science data production environment are COTS provided interfaces. In addition, the operator user interface for the DAAC Quality Assurance operations position will be a ECS custom application. This custom graphical interface will be created with the aid of the Integrated Computer Solutions' Builder Xcessory. Builder Xcessory enables the developer to manage Motif graphical user interface projects by providing a WYSIWYG, drag and drop, visual development environment. Once an interface is constructed, Builder Xcessory will generate C++ code which represents the GUI and encapsulates the C-based Motif Widget set. The generated C++ code can then be combined with other Processing CSCI specific code.

4.7.2.1 Off-The-Shelf Interfaces

The Job Scheduling COTS products, AutoSys and AutoXpert, provide GUIs to interface with their applications. The AutoSys GUI, known as the Operator Console, provides capabilities to manage and monitor a schedule of jobs. The GUI provides visibility to the job stream as well as supporting alarm and monitoring mechanisms. Also provided is a set of job interfaces which allow the operator to interact with a job. These interfaces support job creation, modification, cancellation, suspension, and modification. More information on the AutoSys product can be found in Section 4.1.4 and the underlying subsections.

AutoXpert is another GUI which used in conjunction with AutoXpert can provide three different abstract views of the current production schedule. These three graphical views are called TimeScape, JobScape, and HostScape. All three views offer intuitive GUI controls and full-color to illustrate job status and activity in real-time/projection and simulation modes

TimeScape graphically displays the timing and duration of currently running jobs. Jobs are displayed in a Gannt style ribbon graph that shows both starting and ending times. This type of representation gives you a comprehensive view of how job dependencies affect job duration and how modifications to the current stream of events will affect overall job execution. TimeScape provides real-time monitoring, projection, and simulation. In real-time/projection mode, TimeScape shows real-time execution alongside projections based on past runs. Projections compare the current run to previous runs and provide a prediction of how current events will affect the future. In addition, the duration of a job can be modified and the downstream impact to the job schedule can be determined. Also, the completion status of a job can be changed to determine the impact on dependent jobs. In simulation mode, an entire schedule of jobs can be sped up to determine how the schedule will be followed.

JobScape gives a detailed representation of job flow from a logical, or dependency, point of view.

It depicts jobs in a flow diagram, showing the starting conditions for a each job. The flow of the job steam can be followed visually, step by step. Simulations in JobScape also provide a valuable means for testing jobs before they are activated. By running in simulation mode, the job definition can be checked for errors.

HostScape provides a view of jobs that are currently executing and the resources being used to execute the job. HostScape continually verifies whether AutoSys can start jobs on each machine and notifies the console if a machine is unavailable.

Each of these views can assist the operations staff in providing production management oversight. The capabilities of viewing the job stream abstractly were originally mapped to the Production Management CSC of the Planning CSCI. Because of the capabilities of AutoXpert to provide this capability, these capabilities are now mapped to the Processing CSCI. Additional capabilities to perform what-ifs on the daily job schedule are also provided. These capabilities will be helpful in determining the downstream affects of production anomalies, i.e., a job running longer than predicted or a job failure, would have on the job schedule.

More information on AutoSys and AutoXpert product can be found in Section 4.1.4 and the underlying subsections.

4.7.2.2 Processing CSCI User Interfaces

This section is intended to describe the data that may be displayed for the operations of the Processing CSCI's applications. The exact definition of the GUI will be decided by requesting user suggestions and through demonstrating prototypes.

Quality Assurance Monitor

The Quality Assurance Monitor is a utility which provides the operations staff with the capability to perform quality assurance activities which are defined as data visualization, updating quality assurance metadata, and subscribing to data. The GUI for the Quality Assurance Monitor will be constructed with custom code which interfaces to the PDPS Database for storage of outstanding quality assurance subscriptions. The subscription editor would contain a single view which would allow an operator to enter product subscriptions to the Science Data Server. Two lists could be provided: one list containing unsubscribed products and the second list containing subscribed products. Functions would be provided to assist the operator in adding or deleting subscriptions to these products.

As part of the GUI, utilities would be provided to initiate data visualization tools, such as EOSVIEW, which will be used to visualize a science data product and to update quality assurance metadata for a data product. The update of the quality assurance metadata would occur through the use of public classes provided by the Data Server.

4.7.3 Reports

A variety of ad-hoc and canned reports will be available to the DAAC operations staff to assist in the monitoring of the activities associated with the Processing CSCI. These reports are readily accessible given that the Processing CSCI persistent data is maintained in the PDPS Database, a SYBASE RDBMS. Also, ECS application management information is maintained in the MSS database, which is used to log system events. The canned reports will include the following:

a. Planning Workload and Processing Turn-Around Reports—These reports will provide tracking information on planned vs. actual processing results. The information provided will include job statistics for a Data Processing Request to allow comparisons in planned vs. actual resource consumption, planned start and end time vs. actual start and end time, planned resource, i.e., machine, allocation vs. actual resource, etc.

b. Processing Management Reports—These reports will provide the operations staff information on Processing application software events which have occurred. This information will be available from the MSS database.

c. Job Status/Event Reports—These reports will provide information on the history of the AutoSys job schedule. All status and event changes for a job will be logged in the AutoSys Database. Any information associated with a job can reported on.

d. Resource Allocation Report—These reports will track the ability of the Processing CSCI Resource Management CSC to effectively manage the Science Processing Hardware resources.

e. Quality Assurance Report—These reports will be used to track what products have been Q/A'ed vs. the products awaiting Q/A.

f. DPR Job Status Report—These reports will capture the current state of all the jobs which have been created to support the processing of one PGE.

g. PGE Resource Profile Report—These reports will capture information on the actual usage of resources which have been used by a PGE. This information will be fed back into the PDPS Database to update the current PGE Resource Profile.

Other ad-hoc reports can be defined to assist the Production Planning Operations staff in performing their activities. The PDPS Database is the repository used to maintain information on Production Requests and associated Data Processing Requests, Data Subscriptions, PGE Profiles, etc. These reports can be used to track modifications and provide historical information on these data objects. Because of the use of a consistent RDBMS throughout ECS, the sharing of information between different databases is simplified and will allow for consistent definitions for any number of reports.

5. SDPTK - Science Data Processing Toolkit CSCI

5.1 CSCI Overview

The Science Data Processing Toolkit is on the Incremental Track and has been documented elsewhere. See the following references for SDPTK CSCI details.

333-CD-001-002	SDP Toolkit Users Guide for the ECS Project
193-801-SD4-001	PGS Toolkit Requirements Specification for the ECS Project, FINAL, [AKA GSFC 423-16-02]
175-WK-001-001	Draft HDF-EOS Primer for Version 1 EOSDIS

Table 5.1-1 provides a list of the Computer Software Components in the SDPTK CSCI, indicating which components will be mapped to COTS and which will be custom. EOS-HDF is described in the "Draft HDF-EOS Primer for Version 1 EOSDIS." The remaining CSCs are documented in Section 6 of the "SDP Toolkit Users Guide for the ECS Project" and Section 6 of the "PGS Toolkit Requirements Specification for the ECS Project."

Table 5.1-1. SDPTK Computer Software Components

CSC	Description	Type (Custom = DEV; off-the-shelf = OTS)
Ancillary Data Access	Provides subsetting and parameter searches to several data sets, i.e., NMC, Digital Elevation Model, DIgital CHart of the World.	DEV
Celestial Body Position	Provides position vector of planetary and stellar bodies.	DEV
Coordinate System Conversion	Provides conversion between platform frame and a variety of other reference frames, i.e., Earth Centered Inertial.	DEV
Constant and Unit Conversions	Provides access to a table of standard scientific constants and unit conversions.	DEV
Ephemeris Data Access	Provides access to TRMM and EOS platform ephemeris; simulated or actual.	DEV
Geo Coordinate Transformation	Provides transformation of geocentric lat./long coordinates to a variety of standard projections.	DEV
Input/Output	Provides opens and closes of files through information in a process control file.	DEV
Memory Management	Provides users access to dynamic and shared memory allocation.	DEV
Meta Data Access	Provides access to metadata configuration file (contents of ECS core metadata standard).	DEV
Process Control	Provides access to process control information supplied by the user and by the production system, including file handles and file attributes.	DEV
Status Message File (Error/Status)	Provides access to status message files created by user and by the production system.	DEV
Time Date Conversion	Provides conversions between Toolkit internal time and various standard time systems.	DEV
Math Package	Provides a mathematics and statistics library for science algorithm development and product production.	COTS
Graphics Library	Provides a library of graphics routines for visual display of EOS products and quality data.	COTS
EOS-HDF	Provides an extension to NCSA HDF API - This API includes support for grid, point and swath data structures.	DEV

6. DPREP - Science Data Pre-Processing CSCI

6.1 CSCI Overview

The Science Data Pre-Processing CSCI has been absorbed into the designs of the Processing CSCI of the Data Processing Subsystem and the Ingest CSCI of the Ingest Subsystem. See Section 4 of this document and Section 4 of the Ingest Subsystem volume.

This page intentionally left blank.

7. AITTL - Algorithm I&T CSCI

7.1 CSCI Overview

The purpose of the Algorithm Integration and Test Tools (AITTL) Computer Software Configuration Item (CSCI) is to provide the software tools required to integrate and test (I&T) the science software at the Distributed Active Archive Center (DAAC). The science software will be developed by a Science Computing Facility (SCF), which may be at a different location than the DAAC.

The division of responsibilities between the DAAC and the SCF is generally the following: The SCF is responsible for developing the science software and ensuring that the generated products are scientifically correct. The DAAC is responsible for integrating the science software into the production environment, ensuring that the software will run safely (i.e., will not interfere with the production environment or with other product generation), and running the software in a production mode. The machine on which the science software is developed at the SCF is liable to be of a different class than the machine on which the software is run at the DAAC.

The developer/operator division that is characteristic of the science software lifecycle causes the DAAC I&T personnel to have certain special requirements. The I&T team needs to be able to do the following things:

- Receive a science software delivery from the SCF. The delivery will be made either electronically or on hard media.
- Examine the science software delivery for correctness and completeness. This includes: examining accompanying documentation, verifying that prescribed coding standards have been followed, and running preliminary static and dynamic diagnostic tools to check for potential errors. The delivered files must also be placed under configuration control.
- Compile and link the delivered source files.
- Run test cases. For the most part, these test cases will be supplied by the SCF as part of the delivery. Also supplied by the SCF, for each test case, will be a set of required input files and a corresponding set of output files. Since some of the input files may already reside at the DAAC, the I&T personnel also need the ability to manually stage inputs from the data servers. The DAAC I&T team will re-run each test case and compare their outputs with those supplied by the SCF. Because the SCF and DAAC machines may have different precision, the file comparison utility needs to be more sophisticated than the usual Unix "diff" tool: it needs to be able to screen out differences that are due only to differences in precision. In addition, the ability to examine product metadata is required.
- Diagnose errors. This requires access to: interactive debuggers, screen dump utilities, data visualization tools, and so on.
- Collect resource requirement statistics. This includes: CPU time, memory requirements, disk space requirements, and so on. The collection of such statistics is required both as a "sanity check", to make sure that the measured requirements match the expected values, and also for the PGE database, which is used by the Planning and Processing systems to execute the science software properly.

- Update the system databases once the science software has completed acceptance testing. This includes: adding the source code and documentation to the data server, so that they may be distributed to requesting users, and adding the resource requirement information to the PGE database.
- Write reports and maintain the I&T log.
- Write additional ad-hoc test tools.

Satisfaction of these requirements is distributed across several systems. The ingest client for science software will be supplied by the Ingest Subsystem (INS) of the Science Data Processing Segment (SDPS). Configuration management and problem tracking tools will be provided by the Management Subsystem (MSS) of the Communications and System Management Segment (CSMS). Compilers, linkers, debuggers, and other development and operating system tools will be furnished by the Algorithm Integration and Test Hardware Configuration Item (AITHW) of the Data Processing Subsystem (DPS) of SDPS, since such utilities are so closely wedded to the processing platforms. In addition, some of the standards checking and profiling requirements will probably be satisfied by AITHW as well, since certain of these capabilities will be found in compilers and development environments. The remaining requirements will be satisfied by AITTL.

The AITTL-supplied tools therefore fall into the following categories:

- Tools to view science software documentation.
- Tools to check compliance of science software to ESDIS-specified coding standards.
- Code analysis tools.
- Data visualization tools.
- HDF file comparison tool.
- Binary file comparison environment.
- Profiling tool.
- Tool to register science software files in the processing system.
- Tool to kickoff execution of PGEs.
- Tools for writing reports and maintaining the I&T logs.
- Tools for checking Process Control Files and for prohibited functions.
- Tools to display product metadata.

Most of the functions for the Algorithm Integration & Test CSCI are being developed as working prototypes and delivered for Interim Release 1(IR-1). The release A design approach is based on using the IR-1 release as a basis for which to add Release A capabilities. Since AITTL CSCI is very operations oriented, as much feedback from DAAC Operations who have used the IR-1 AITTL tools, will be incorporated into the Release A AITTL capabilities. As in IR-1, the Job Scheduler COTS product, AutoSys, can be used to manage the execution of jobs on AITTL HW. These jobs represent the execution of a PGE or associated preparation or post-processing jobs. More information is included on AutoSys in Section 4 of this document. Included in the previously provided information is material related to AutoSys' Operations interfaces. Presented as part of the design are object model representations of the GUI interfaces being developed for AITTL.

The formation of the AITTL design material is the following:

a. Section 7.2—AITTL Context.

b. Section 7.3—AITTL Object Model.

c. Section 7.4—AITTL Dynamic Model focussing on AIT Manager GUI scenarios, including some detailed event traces.

d. Section 7.5—AITTLlFunctional Model representing the many tool dependent data flows for AITTL.

f. Section 7.6—AITTL Operational Scenarios, presents some operational scenarios showing how the AITTL tools may be used in the integration and test process, rather than the usual dynamic model;

g. Section 7.7—AITTL Structure which defines the CSC components of the AITTL CSCI.

h. Section 7.8—AITTL Management and Operation defines some management and operations concepts used in developing the AITTL CSCI design.

For additional information on science software integration and test procedures, see also: 205-CD-002-001, Science User's Guide and Operations Procedures Handbook for the ECS Project, Part 4, and JU9403V1, Science Software Integration and Test. For information on the ESDIS science software coding standards and guidelines, see: 423-16-01, Data Production Software and Science Computing Facility (SCF) Standards and Guidelines.

7.2 CSCI Context

The context diagram for the AITTL CSCI is shown in Figure 7.2-1.

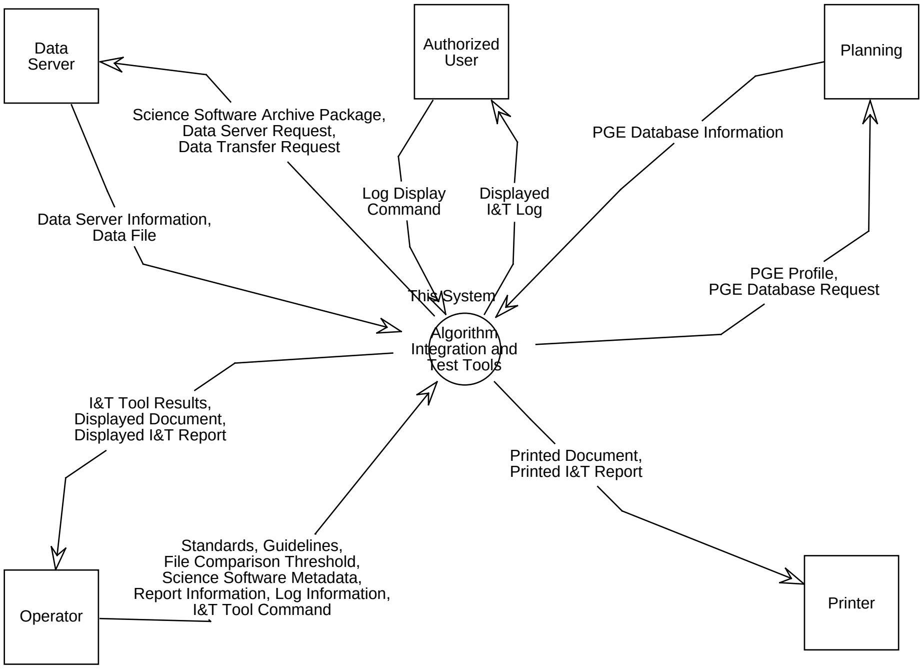
AITTL has interfaces with two other ECS subsystems: the Data Server Subsystem and the Planning Subsystem.

The purpose of the interface with the Data Server is to archive a tested delivery *(Science Software Archive Package)* and to manually stage inputs *(Data File)*. The Data Server also provides information about science software that is already archived on the data server *(Data Server Information)*. Requests for data transfers *(Data Transfer Request)* and other information *(Data Server Request)* are sent to the Data Server.

The purpose of the interface with Planning is to update the PGE database, a database containing resource requirements and other information about each product generation executive (PGE) of the science software. This information is needed by the system to plan and execute the PGEs correctly and efficiently. Information about PGEs that are already recorded in the database *(PGE Database Information)* are received from Planning. The required information about a PGE (PGE Profile) is sent to Planning to be placed in the database, as are requests to update or get information from the database *(PGE Database Request)*.

The primary interface for AITTL is with the operator(s) responsible for integration and test. The operator sends commands *(I&T Tool Command)* to the various tools, supplies the standards checkers with the desired standards *(Standards)* and guidelines *(Guidelines)*, supplies the file comparison utility with thresholds *(File Comparison Threshold)* to filter out differences due to precision differences, provides metadata *(Science Software Metadata)* about the science software delivery to the utility that updates the data server with the new science software, and enters information into reports *(Report Information)* and into the integration and test log *(Log Information)*. The tools display various results *(I&T Tool Results)*, science software documentation *(Displayed Document)*, and integration and test reports *(Displayed I&T Report)* for the operator to examine.

Finally, the tools send hardcopy of science software documentation *(Printed Document)*, as well as integration and test reports and tool results *(Printed I&T Report)* to the printer.



Data Server
Authorized User
Planning
Science Software Archive Package, Data Server Request, Data Transfer Request
PGE Database Information
Log Display Command
Displayed I&T Log
Data Server Information, Data File
PGE Profile, PGE Database Request
This System
Algorithm Integration and Test Tools
I&T Tool Results, Displayed Document, Displayed I&T Report
Printed Document, Printed I&T Report
Standards, Guidelines, File Comparison Threshold, Science Software Metadata, Report Information, Log Information, I&T Tool Command
Operator
Printer

Figure 7.2-1. Algorithm Integration and Test Tools Context Diagram

7.3 CSCI Object Model

This section gives the object model for the AIT Manager GUI. This GUI is used by the DAAC operator to check in and verify the science software code as delivered by the SCFs. The GUI runs instrument-specific compilation and execution scripts, configuration management scripts, custom code checking, file display and comparison tools, and COTS tools such as office automation and analysis environment programs. The AIT Manager GUI contains a graphical checklist of AI&T steps in delivery and testing of science software, and a display of a log file.

Figure 7.3-1 shows the algorithm integration and test IR-1 object model. Figure 7.3-2 shows the algorithm integration and test support for data sever I/F object model.

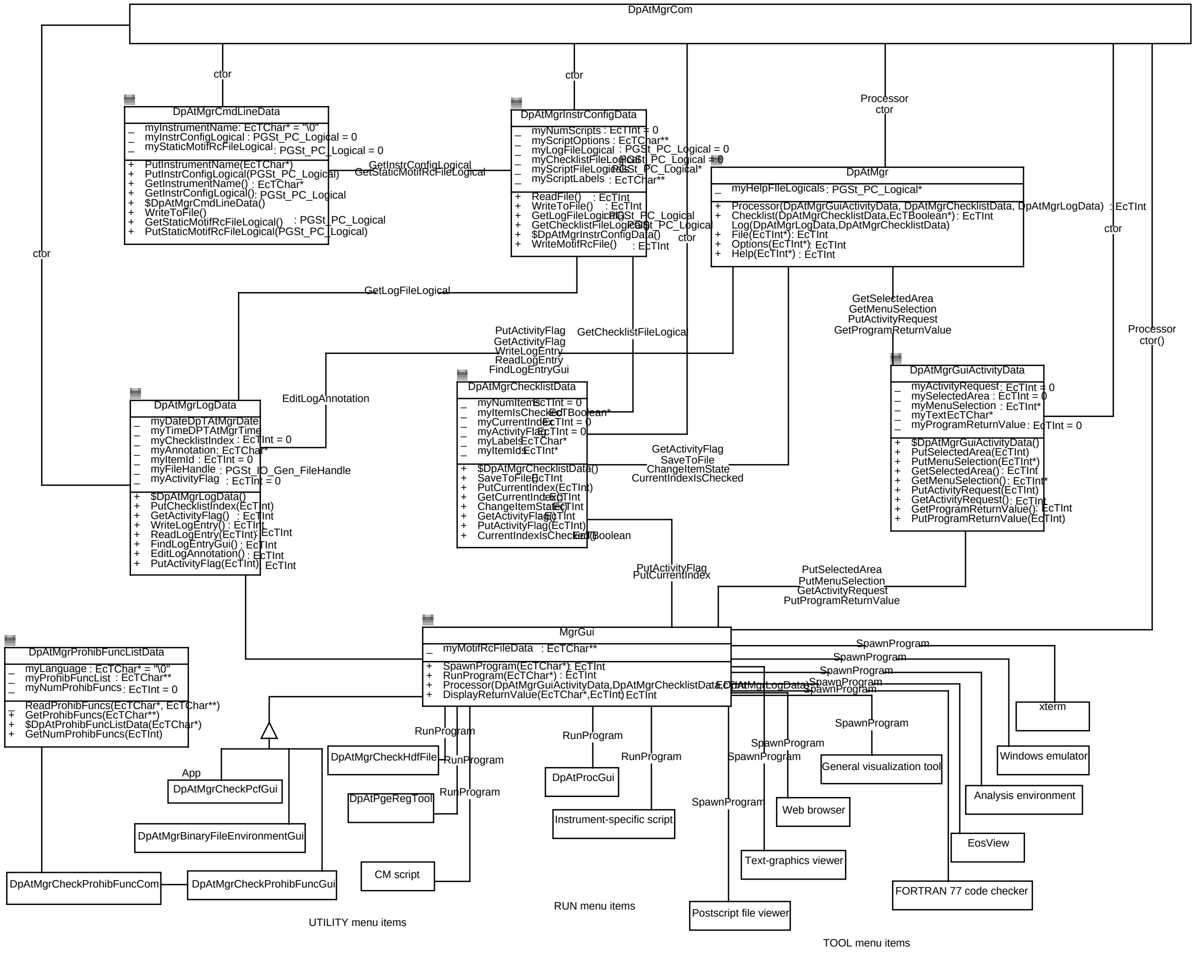

DpAtMgrCom
ctor
ctor
Processor
ctor
DpAtMgrCmdLineData
myInstrumentName: EcTChar* = "\0"
myInstrConfigLogical : PGSt_PC_Logical = 0
myStaticMotifRcFileLogical : PGSt_PC_Logical = 0
PutInstrumentName(EcTChar*)
PutInstrConfigLogical(PGSt_PC_Logical)
GetInstrumentName() : EcTChar*
GetInstrConfigLogical(): PGSt_PC_Logical
$DpAtMgrCmdLineData()
WriteToFile()
GetStaticMotifRcFileLogical() : PGSt_PC_Logical
PutStaticMotifRcFileLogical(PGSt_PC_Logical)
GetInstrConfigLogical
GetStaticMotifRcFileLogical
DpAtMgrInstrConfigData
myNumScripts : EcTInt = 0
myScriptOptions : EcTChar**
myLogFileLogical : PGSt_PC_Logical = 0
myChecklistFileLogical : PGSt_PC_Logical = 0
myScriptFileLogical : PGSt_PC_Logical*
myScriptLabels : EcTChar**
ReadFile() : EcTInt
WriteToFile() : EcTInt
GetLogFileLogical() : PGSt_PC_Logical
GetChecklistFileLogical() : PGSt_PC_Logical
$DpAtMgrInstrConfigData()
WriteMotifRcFile() : EcTInt
DpAtMgr
myHelpFileLogicals : PGSt_PC_Logical*
Processor(DpAtMgrGuiActivityData, DpAtMgrChecklistData, DpAtMgrLogData) : EcTInt
Checklist(DpAtMgrChecklistData,EcTBoolean*) : EcTInt
Log(DpAtMgrLogData,DpAtMgrChecklistData)
File(EcTInt*) : EcTInt
Options(EcTInt*) : EcTInt
Help(EcTInt*) : EcTInt
ctor
ctor
ctor
GetLogFileLogical
PutActivityFlag
GetActivityFlag
WriteLogEntry
ReadLogEntry
FindLogEntryGui
GetChecklistFileLogical
GetSelectedArea
GetMenuSelection
PutActivityRequest
GetProgramReturnValue
Processor
ctor()
EditLogAnnotation
DpAtMgrLogData
myDateDpTAtMgrDate
myTimeDPTAtMgrTime
myChecklistIndex : EcTInt = 0
myAnnotation: EcTChar*
myItemId : EcTInt = 0
myFileHandle : PGSt_IO_Gen_FileHandle
myActivityFlag : EcTInt = 0
$DpAtMgrLogData()
PutChecklistIndex(EcTInt)
GetActivityFlag() : EcTInt
WriteLogEntry() : EcTInt
ReadLogEntry(EcTInt) : EcTInt
FindLogEntryGui() : EcTInt
EditLogAnnotation() : EcTInt
PutActivityFlag(EcTInt) : EcTInt
DpAtMgrChecklistData
myNumItems : EcTInt = 0
myItemIsChecked : EcTBoolean*
myCurrentIndex : EcTInt = 0
myActivityFlag : EcTInt = 0
myLabels : EcTChar*
myItemIds : EcTInt*
$DpAtMgrChecklistData()
SaveToFile() : EcTInt
PutCurrentIndex(EcTInt)
GetCurrentIndex() : EcTInt
ChangeItemState() : EcTInt
GetActivityFlag() : EcTInt
PutActivityFlag(EcTInt)
CurrentIndexIsChecked() : EcTBoolean
GetActivityFlag
SaveToFile
ChangeItemState
CurrentIndexIsChecked
DpAtMgrGuiActivityData
myActivityRequest : EcTInt = 0
mySelectedArea : EcTInt = 0
myMenuSelection : EcTInt*
myTextEcTChar*
myProgramReturnValue : EcTInt = 0
$DpAtMgrGuiActivityData()
PutSelectedArea(EcTInt)
PutMenuSelection(EcTInt*)
GetSelectedArea() : EcTInt
GetMenuSelection() : EcTInt*
PutActivityRequest(EcTInt)
GetActivityRequest() : EcTInt
GetProgramReturnValue() : EcTInt
PutProgramReturnValue(EcTInt)
PutActivityFlag
PutCurrentIndex
PutSelectedArea
PutMenuSelection
GetActivityRequest
PutProgramReturnValue
DpAtMgrProhibFuncListData
myLanguage : EcTChar* = "\0"
myProhibFuncList : EcTChar**
myNumProhibFuncs : EcTInt = 0
ReadProhibFuncs(EcTChar*, EcTChar**)
GetProhibFuncs(EcTChar**)
$DpAtProhibFuncListData(EcTChar*)
GetNumProhibFuncs(EcTInt)
MgrGui
myMotifRcFileData : EcTChar**
SpawnProgram(EcTChar*) : EcTInt
RunProgram(EcTChar*) : EcTInt
Processor(DpAtMgrGuiActivityData,DpAtMgrChecklistData,DpAtMgrLogData) : EcTInt
DisplayReturnValue(EcTChar*,EcTInt) : EcTInt
SpawnProgram
SpawnProgram
SpawnProgram
SpawnProgram
SpawnProgram
SpawnProgram
SpawnProgram
SpawnProgram
SpawnProgram
RunProgram
RunProgram
RunProgram
RunProgram
RunProgram
App
DpAtMgrCheckPcfGui
DpAtMgrBinaryFileEnvironmentGui
DpAtMgrCheckProhibFuncCom
DpAtMgrCheckProhibFuncGui
DpAtMgrCheckHdfFile
DpAtPgeRegTool
CM script
UTILITY menu items
DpAtProcGui
Instrument-specific script
RUN menu items
xterm
Windows emulator
General visualization tool
Analysis environment
Web browser
EosView
Text-graphics viewer
FORTRAN 77 code checker
Postscript file viewer
TOOL menu items

Figure 7.3-1. Algorithm Integration and Test IR-1 Object Model

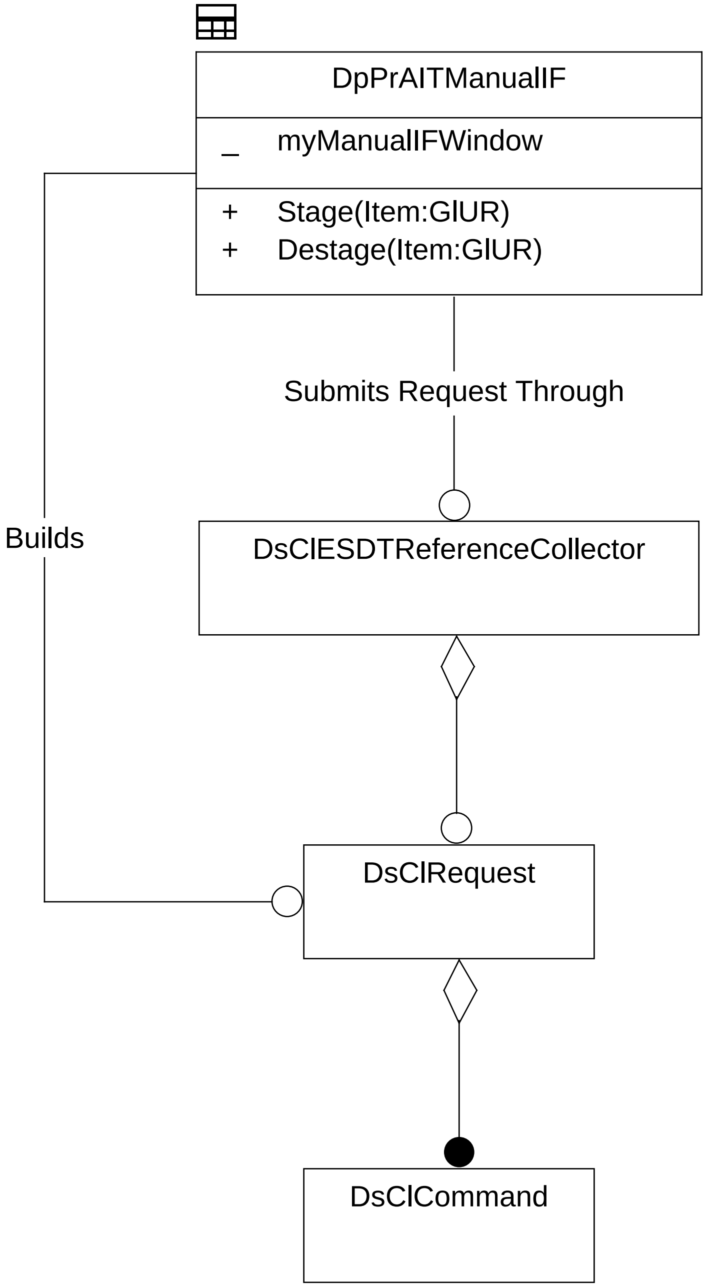


Figure 7.3-2. Algorithm Integration and Test Support for Data Server I/F Object Model

7.3.1 Analysisenvironment Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:
This is an Abstract Class. It represents the following tools, SPARWORKS on a Sun machine or CASEVision for an SGI machinge. This association is used to show how the tool can be initiated from the GUI. It is callable from the UNIX command line and initiates analysis environment.

Attributes:

None

Operations:

None

Associations:

The Analysisenvironment class has associations with the following classes:
Class: MgrGui SpawnProgram

7.3.2 CMscript Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This is an Abstract Class. It represents a Script for use with CM tool (ClearCase). It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The CMscript class has associations with the following classes:
Class: MgrGui RunProgram

7.3.3 DpAtMgr Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

General processor of AIT Manager Kicks off COTS, custom and instrument-specific scripts Calls checklist, log, other processors

Attributes:

myHelpFIleLogicals - PCF file logicals for Help files to display

Data Type: PGSt_PC_Logical*

Privilege: Private

Default Value:

Operations:

Checklist - Checklist processor Called by DpAtMgr.Processor() Input: DpAtMgrChecklistData instance

Arguments: DpAtMgrChecklistData,EcTBoolean*

Return Type: EcTInt

Privilege: Public

File - File menu processor Called by DpAtMgr.Processor() Input: menuSelection, ...

Arguments: EcTInt*

Return Type: EcTInt

Privilege: Public

Help - Help menu processor

Arguments: EcTInt*

Return Type: EcTInt

Privilege: Public

Log - Log processor   Called by DpAtMgr.Processor()

Arguments: DpAtMgrLogData,DpAtMgrChecklistData

Options - Options menu processor Called by DpAtMgr.Processor() Input: menuSelection, ...

Arguments: EcTInt*

Return Type: EcTInt

Privilege: Public

Processor - Main processor for data returned by GUI
Arguments: DpAtMgrGuiActivityData, DpAtMgrChecklistData, DpAtMgrLogData
Return Type: EcTInt
Privilege: Public

Associations:

The DpAtMgr class has associations with the following classes:
Class: DpAtMgrChecklistData
GetActivityFlagSaveToFileChangeItemStateCurrentIndexIsChecked
Class: DpAtMgrGuiActivityData
GetSelectedAreaGetMenuSelectionPutActivityRequestGetProgramReturnValue
Class: DpAtMgrCom Processorctor
Class: DpAtMgrLogData
PutActivityFlagGetActivityFlagWriteLogEntryReadLogEntryFindLogEntryGuiEditLog
Annotation

7.3.4 DpAtMgrBinaryFileEnvironmentGui Class

Parent Class: MgrGui
Public: NoDistributed Object: No
Purpose and Description:

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DpAtMgrBinaryFileEnvironmentGui class has associations with the following classes:
None

7.3.5 DpAtMgrCheckHdfFile Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This is an Abstract Class. It is used to represent the IDL program to compare two HDF files, and also to display metadata. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The DpAtMgrCheckHdfFile class has associations with the following classes:

Class: MgrGui RunProgram

7.3.6 DpAtMgrCheckPcfGui Class

Parent Class: MgrGui

Public: No Distributed Object: No

Purpose and Description:

DpAtMgrCheckPcfGui is the GUI for checking Process Control Files (PCFs) for valid syntax and required contents.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DpAtMgrCheckPcfGui class has associations with the following classes:

None

7.3.7 DpAtMgrCheckProhibFuncCom Class

Parent Class: Not Applicable

Public: NoDistributed Object: No

Purpose and Description:

This is an Abstract Class used to represent a.Prohibited function checker. This is the Unix command line version

Attributes:

None

Operations:

None

Associations:

The DpAtMgrCheckProhibFuncCom class has associations with the following classes:

Class: DpAtMgrCheckProhibFuncGui

Class: DpAtMgrProhibFuncListData

7.3.8 DpAtMgrCheckProhibFuncGui Class

Parent Class: MgrGui

Public: No Distributed Object: No

Purpose and Description:

Input GUI for prohibited function checker

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DpAtMgrCheckProhibFuncGui class has associations with the following classes:

Class: DpAtMgrCheckProhibFuncCom

7.3.9 DpAtMgrChecklistData Class

Parent Class: Not Applicable

Public: NoDistributed Object: No

Purpose and Description:

Stores data for AIT Manager checklist

Attributes:

myActivityFlag - Flag to indicate activity received from GUI =0, No activity =1, myCurrentIndex changed state selected/not selected =2, "Save Checklist" button pushed
Data Type: EcTInt
Privilege: Private
Default Value: 0

myCurrentIndex - Current index of checklist item If myActivityFlag=1 received from GUI, this item changed state from/to checked/unchecked
Data Type: EcTInt
Privilege: Private
Default Value: 0

myItemIds - Array of checklist identifiers, numbered consecutively One for each checklist item
Data Type: EcTInt*
Privilege: Private
Default Value:

myItemIsChecked - Array of states for checklist items =EcDFalse, item is not checked =EcDTrue, item is checked
Data Type: EcTBoolean*
Privilege: Private
Default Value:

myLabels - Array of checklist labels to display on GUI screen One label for each checklist item
Data Type: EcTChar*
Privilege: Private
Default Value:

myNumItems - Number of items for this checklist
Data Type: EcTInt
Privilege: Private
Default Value: 0

Operations:

$DpAtMgrChecklistData - Constructor for class DpAtMgrChecklist data Reads checklist data from file
Arguments:
Return Type: Void
Privilege: Public

ChangeItemState - Changes state myItemIsChecked[myCurrentIndex] from/to checked/unchecked
Arguments:
Return Type: EcTInt
Privilege: Public

CurrentIndexIsChecked - Returns EcDTrue if myCurrentIndex is checked; EcDFalse if not
Arguments:
Return Type: EcTBoolean
Privilege: Public

GetActivityFlag - Get value of myActivityFlag
Arguments:
Return Type: EcTInt
Privilege: Public

GetCurrentIndex - Gets index (ID) of current checklist item
Arguments:
Return Type: EcTInt
Privilege: Public

PutActivityFlag - Set value of myActivityFlag
Arguments: EcTInt
Return Type: Void
Privilege: Public

PutCurrentIndex - Sets index (ID) of current checklist item
Arguments: EcTInt
Return Type: Void
Privilege: Public

SaveToFile - Saves checklist data to a file
Arguments:
Return Type: EcTInt
Privilege: Public

Associations:

The DpAtMgrChecklistData class has associations with the following classes:
Class: DpAtMgr GetActivityFlagSaveToFileChangeItemStateCurrentIndexIsChecked
Class: DpAtMgrInstrConfigData GetChecklistFileLogical
Class: MgrGui PutActivityFlagPutCurrentIndex
Class: DpAtMgrCom ctor

7.3.10 DpAtMgrCmdLineData Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
Stores all data which may be specified on the initial AIT Manager command line.

Attributes:

myInstrConfigLogical - Logical ID in Process Control File (PCF) for instrument configuration data for this AIT Manager session
Data Type: PGSt_PC_Logical
Privilege: Private
Default Value: 0

myInstrumentName - Name of instrument for this AIT Manager session
Data Type: EcTChar*
Privilege: Private
Default Value: "\0"

myStaticMotifRcFileLogical - PCF file logical for user-editable static Motif menu data for this AIT Manager session *****This file is in the same format as the Motif resources file It contains all menu labels, sub-menu labels, etc. *except* the Run menu data, plus any user preferences such as colrs, fonts, etc.
Data Type: PGSt_PC_Logical
Privilege: Private
Default Value: 0

Operations:

$DpAtMgrCmdLineData - Constructor for DpAtMgrCmdLineData
Arguments:
Return Type: Void

Privilege: Public

GetInstrConfigLogical - Gets instrument configuration file logical ID for PCF
Arguments:
Return Type: PGSt_PC_Logical
Privilege: Public

GetInstrumentName - Gets instrument name
Arguments:
Return Type: EcTChar*
Privilege: Public

GetStaticMotifRcFileLogical - Gets static motif resources file logical for PCF
Arguments:
Return Type: PGSt_PC_Logical
Privilege: Public

PutInstrConfigLogical
Arguments: PGSt_PC_Logical
Return Type: Void
Privilege: Public

PutInstrumentName - Puts instrument name into storage
Arguments: EcTChar*
Return Type: Void
Privilege: Public

PutStaticMotifRcFileLogical
Arguments: PGSt_PC_Logical
Return Type: Void
Privilege: Public

WriteToFile - Writes command line configuration for this AIT Manager session to file
Arguments:
Return Type: Void
Privilege: Public

Associations:

The DpAtMgrCmdLineData class has associations with the following classes:
Class: DpAtMgrInstrConfigData GetInstrConfigLogicalGetStaticMotifRcFileLogical
Class: DpAtMgrCom ctor

7.3.11 DpAtMgrCom Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This is an Abstract Class used to represent the Main program module for invoking the AIT Manager. It is the main program, callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The DpAtMgrCom class has associations with the following classes:

Class: MgrGui Processorctor()
Class: DpAtMgr Processorctor
Class: DpAtMgrChecklistData ctor
Class: DpAtMgrCmdLineData ctor
Class: DpAtMgrGuiActivityData ctor
Class: DpAtMgrInstrConfigData ctor
Class: DpAtMgrLogData ctor

7.3.12 DpAtMgrGuiActivityData Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class is the interface between the GUI/Motif code and the external code. GUI callbacks are accesses of this class's data through its operations.

Attributes:

myActivityRequest - GUI activity requested by calling module =0, take no action =1, redisplay entire GUI =2, redisplay checklist only =3, redisplay log only =4, destroy entire GUI =5, edit current log entry

Data Type: EcTInt

Privilege: Private

Default Value: 0

myMenuSelection - Index of item selected on menus myMenuSelection[0] = main menu item index myMenuSelection[0]=0, File menu myMenuSelection[0]=1, Options menu myMenuSelection[0]=2, Tools menu myMenuSelection[0]=3, Run menu myMenuSelection[0]=4, Utilities menu myMenuSelection[0]=5, Help menu myMenuSelection[1] = sub menu item index myMenuSelection[1]=0, 1st sub menu item etc myMenuSelection[2] = sub sub menu item index myMenuSelection[2]=0, 1st sub sub menu item
Data Type: EcTInt*
Privilege: Private
Default Value:

myProgramReturnValue - Return value from program selected from Tools, Run or Help menu
Data Type: EcTInt
Privilege: Private
Default Value: 0

mySelectedArea - Portion of GUI that the user clicked on =0, no selection =1, main menu =2, checklist =3, log
Data Type: EcTInt
Privilege: Private
Default Value: 0

myText - Text written by user into GUI Used to annotate log
Data Type: EcTChar*
Privilege: Private
Default Value:

Operations:

$DpAtMgrGuiActivityData - Constructor for DpAtMgrGuiActivityData
Arguments:
Return Type: Void
Privilege: Public

GetActivityRequest - Get value of myActivityRequest
Arguments:
Return Type: EcTInt
Privilege: Public

GetMenuSelection - Get value of myMenuSelection
Arguments:
Return Type: EcTInt*
Privilege: Public

GetProgramReturnValue - Get value of myProgramReturnValue
Arguments:
Return Type: EcTInt
Privilege: Public

GetSelectedArea - Get value of mySelectedArea
Arguments:
Return Type: EcTInt
Privilege: Public

PutActivityRequest - Set value of myActivityRequest
Arguments: EcTInt
Return Type: Void
Privilege: Public

PutMenuSelection - Set value of myMenuSelection
Arguments: EcTInt*
Return Type: Void
Privilege: Public

PutProgramReturnValue - Set value of myProgramReturnValue
Arguments: EcTInt
Return Type: Void
Privilege: Public

PutSelectedArea - Set value of mySelectedArea
Arguments: EcTInt
Return Type: Void
Privilege: Public

Associations:

The DpAtMgrGuiActivityData class has associations with the following classes:
Class: DpAtMgr GetSelectedAreaGetMenuSelectionPutActivityRequestGetProgramReturnValue
Class: MgrGui PutSelectedAreaPutMenuSelectionGetActivityRequestPutProgramReturnValue
Class: DpAtMgrCom ctor

7.3.13 DpAtMgrInstrConfigData Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

Stores instrument-specific configuration data

Attributes:

myChecklistFileLogical - Checklist file PCF logical for this instrument configuration
Data Type: PGSt_PC_Logical
Privilege: Private
Default Value: 0

myLogFileLogical - Log file PCF logical for this instrument configuration
Data Type: PGSt_PC_Logical
Privilege: Private
Default Value: 0

myNumScripts - Number of scripts available for this instrument configuration
Data Type: EcTInt
Privilege: Private
Default Value: 0

myScriptFileLogicals - Script file PCF logicals for this instrument configuration, one for each script
Data Type: PGSt_PC_Logical*
Privilege: Private
Default Value:

myScriptLabels - Script labels for this instrument configuration, for display on GUI menu, one for each script
Data Type: EcTChar**
Privilege: Private
Default Value:

myScriptOptions - Command line options available for this instrument configuration, one for each script
Data Type: EcTChar**
Privilege: Private
Default Value:

Operations:

$DpAtMgrInstrConfigData - Constructor for DpAtMgrInstrConfigData
Arguments:
Return Type: Void
Privilege: Public

GetChecklistFileLogical - Gets myChecklistFileLogical
Arguments:
Return Type: PGSt_PC_Logical
Privilege: Public

GetLogFileLogical - Gets myLogFileLogical
Arguments:
Return Type: PGSt_PC_Logical
Privilege: Public

ReadFile - Read an instrument configuration file to set all the private data members of this class
Arguments:
Return Type: EcTInt
Privilege: Public

WriteMotifRcFile - Reads the static Motif resources file, and writes its data and the Run/ script data to the dynamic Motif resources file
Arguments:
Return Type: EcTInt
Privilege: Public

WriteToFile - Write all private data members of this class to an instrument configuration file
Arguments:
Return Type: EcTInt
Privilege: Public

Associations:

The DpAtMgrInstrConfigData class has associations with the following classes:
Class: DpAtMgrChecklistData GetChecklistFileLogical
Class: DpAtMgrCmdLineData GetInstrConfigLogicalGetStaticMotifRcFileLogical
Class: DpAtMgrLogData GetLogFileLogical
Class: DpAtMgrCom ctor

7.3.14 DpAtMgrLogData Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

Stores data for AIT Manager instrument-specific log

Attributes:

myActivityFlag - Flag to indicate activity received from GUI =0, No activity =1, Checklist item was checked =2, "NEXT" button pushed =3, "CANCEL" button pushed =4, "LAST" button pushed

Data Type: EcTInt

Privilege: Private

Default Value: 0

myAnnotation - Annotation text for this log entry

Data Type: EcTChar*

Privilege: Private

Default Value:

myChecklistIndex - Index of checklist item which generated this log entry

Data Type: EcTInt

Privilege: Private

Default Value: 0

myDate - Date of this log entry

Data Type: DpTAtMgrDate

Privilege: Private

Default Value:

myFileHandle - Log file handle for input to read/write functions

Data Type: PGSt_IO_Gen_FileHandle

Privilege: Private

Default Value:

myItemId - Unique log entry ID

Data Type: EcTInt

Privilege: Private

Default Value: 0

myTime - Time of this log entry

Data Type: DPTAtMgrTime

Privilege: Private

Default Value:

Operations:

$DpAtMgrLogData - DpAtMgrLogData constructor Reads last entry in log file
Arguments:
Return Type: Void
Privilege: Public

EditLogAnnotation - Edits or creates a log entry annotation is a text editor window
Arguments:
Return Type: EcTInt
Privilege: Public

FindLogEntryGui - GUI for searching for a text string in the log file entries (not the annotations)
Arguments:
Return Type: EcTInt
Privilege: Public

GetActivityFlag - Get value of myActivity flag
Arguments:
Return Type: EcTInt
Privilege: Public

PutActivityFlag
Arguments: EcTInt
Return Type: EcTInt
Privilege: Public

PutChecklistIndex - Sets value of myChecklistIndex
Arguments: EcTInt
Return Type: Void
Privilege: Public

ReadLogEntry - Read a log entry from the log file
Arguments: EcTInt
Return Type: EcTInt
Privilege: Public

WriteLogEntry - Write a log entry to the log file
Arguments:
Return Type: EcTInt
Privilege: Public

Associations:

The DpAtMgrLogData class has associations with the following classes:
Class: MgrGui
Class: DpAtMgrInstrConfigData GetLogFileLogical
Class: DpAtMgr PutActivityFlagGetActivityFlagWriteLogEntryReadLogEntryFindLogEntryGuiEditLog Annotation
Class: DpAtMgrCom ctor

7.3.15 DpAtMgrProhibFuncListData Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
Stores data for prohibited function checker

Attributes:

myLanguage - Computer language of code to check: C, Fortran 77, Fortran 90 or Ada
Data Type: EcTChar*
Privilege: Private
Default Value: "\0"

myNumProhibFuncs - Number of prohibited functions for this language
Data Type: EcTInt
Privilege: Private
Default Value: 0

myProhibFuncList - Array of names of prohibited functions for a given language
Data Type: EcTChar**
Privilege: Private
Default Value:

Operations:

$DpAtProhibFuncListData - Constructor
Arguments: EcTChar*
Return Type: Void
Privilege: Public

GetNumProhibFuncs - Gets number of prohibited functions from this class
Arguments: EcTInt

Return Type: Void
Privilege: Public

GetProhibFuncs - Gets prohibited function list from this class storage
Arguments: EcTChar**
Return Type: Void
Privilege: Public

ReadProhibFuncs - Reads prohibited function list from a file
Arguments: EcTChar*, EcTChar**
Return Type: Void
Privilege: Private

Associations:

The DpAtMgrProhibFuncListData class has associations with the following classes:
Class: DpAtMgrCheckProhibFuncCom

7.3.16 DpAtPgeRegTool Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This is an Abstract Class which represents the GUI for registering a PGE in the Data Processing subsystem. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The DpAtPgeRegTool class has associations with the following classes:
Class: MgrGui RunProgram

7.3.17 DpAtProcGui Class

Parent Class: Not Applicable
Public: No Distributed Object: No

Purpose and Description:
This is an Abstract Class used to represent the .GUI for starting a job in the Data Processing subsystem. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The DpAtProcGui class has associations with the following classes:
Class: MgrGui RunProgram

7.3.18 DpPrAITManualIF Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the Algorithm Integration & Test interface for the manual staging and destaging of data, algorithms, executables, or other stored items which have Universal References.

Attributes:

myManualIFWindow - This represents the GUI which will allow the AI&T position to manually stage and destage data products, executables, algorithms, etc.
Data Type:
Privilege: Private
Default Value:

Operations:

Destage - This operation is used by the Algorithm Integration & Test position to destage a piece of data, a PGE, an algorithm, or anything else which can be located by a UR.
Arguments: Item:GlUR
Return Type: Void
Privilege: Public

Stage - This operation stages the item located by the input UR. This can be a data granule, a PGE, an algorithm, or anything else located by a UR.
Arguments: Item:GlUR
Return Type: Void
Privilege: Public

Associations:

The DpPrAITManualIF class has associations with the following classes:
Class: DsClRequest Builds
Class: DsClESDTReferenceCollector SubmitsRequestThrough

7.3.19 DsClCommand Class

A Data Server Public Class. See Data Server Detailed Design Specification for more details.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DsClCommand class has associations with the following classes:
DsClRequest (Aggregation)

7.3.20 DsClESDTReferenceCollector Class

A Data Server Public Class. See Data Server Detailed Design Specification for more details.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DsClESDTReferenceCollector class has associations with the following classes:
Class: DpPrAITManualIF SubmitsRequestThrough

7.3.21 DsClRequest Class

A Data Server Public Class. See Data Server Detailed Design Specification for more details.

Parent Class: Not Applicable

Attributes:

None

Operations:

None

Associations:

The DsClRequest class has associations with the following classes:
Class: DpPrAITManualIF Builds
DsClESDTReferenceCollector (Aggregation)

7.3.22 EosView Class

An Abstract class used to represent the EOSVIEW Data visualization tool.

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
EosView program; displays contents of ECS HDF files. THIS IS NOT A CLASS. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The EosView class has associations with the following classes:
Class: MgrGui SpawnProgram

7.3.23 FORTRAN77codechecker Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This is an Abstract Class used to represent the .FORTRAN 77 code checker FORCHECK. A FORTRAN 77 code checker is necessary because most compilers (and presumably delivered F77 code) will not adhere strictly to the F77 ANSI standard. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The FORTRAN77codechecker class has associations with the following classes:
Class: MgrGui SpawnProgram

7.3.24 Generalvisualizationtool Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This is an Abstract Class used to represent the General data visualiztion tool IDL. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The Generalvisualizationtool class has associations with the following classes:
Class: MgrGui SpawnProgram

7.3.25 Instrument-specificscript Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This is an Abstract Class used to represent Scripts written by users specific to an instrument configuration. This is used for code check-in, compiling, running, etc. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The Instrument-specificscript class has associations with the following classes:
Class: MgrGui RunProgram

7.3.26 MgrGui Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
MgrGui is the main AIT Manager Gui. This class is necessarily ill-defined since it will

contain code generated by the GUI builder.

Attributes:

myMotifRcFileData - Data read from Motif resources file, including menu labels, and program names and arguments where appropriate
Data Type: EcTChar**
Privilege: Private
Default Value:

Operations:

DisplayReturnValue - Displays value returned by system call on GUI
Arguments: EcTChar*,EcTInt
Return Type: EcTInt
Privilege: Public

Processor - Main AIT Manager GUI processor
Arguments: DpAtMgrGuiActivityData,DpAtMgrChecklistData,DpAtMgrLogData
Return Type: EcTInt
Privilege: Public

RunProgram - Run a program which is callable from the Unix command line Input:Unix program name and options as a single string *****PDL***** Make system call to run program Wait for program to execute Return same return value from program
Arguments: EcTChar*
Return Type: EcTInt
Privilege: Public

SpawnProgram - Spawns a Unix program or script. Immediately returns a return value without waiting for program to execute.
Arguments: EcTChar*
Return Type: EcTInt
Privilege: Public

Associations:

The MgrGui class has associations with the following classes:
Class: DpAtMgrLogData
Class: DpAtMgrCom Processorctor()
Class: DpAtMgrChecklistData PutActivityFlagPutCurrentIndex
Class: DpAtMgrGuiActivityData PutSelectedAreaPutMenuSelectionGetActivityRequestPutProgramReturnValue
Class: CMscript RunProgram

Class: DpAtMgrCheckHdfFile RunProgram
Class: DpAtPgeRegTool RunProgram
Class: DpAtProcGui RunProgram
Class: Instrument-specificscript RunProgram
Class: Analysisenvironment SpawnProgram
Class: EosView SpawnProgram
Class: FORTRAN77codechecker SpawnProgram
Class: Generalvisualizationtool SpawnProgram
Class: Postscriptfileviewer SpawnProgram
Class: Text-graphicsviewer SpawnProgram
Class: Webbrowser SpawnProgram
Class: Windowsemulator SpawnProgram
Class: xterm SpawnProgram

7.3.27 Postscriptfileviewer Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This is an Abstract Class used to represent the .PostScript file viewer Ghostview. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The Postscriptfileviewer class has associations with the following classes:
Class: MgrGui SpawnProgram

7.3.28 Text-graphicsviewer Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This is an abstract class used to represent the Text and graphics viewer Adobe Acrobat. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The Text-graphicsviewer class has associations with the following classes:
Class: MgrGui SpawnProgram

7.3.29 Webbrowser Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This is an abstract class used to represent the WorldWideWeb browser Mosaic.. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The Webbrowser class has associations with the following classes:
Class: MgrGui SpawnProgram

7.3.30 Windowsemulator Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This is an abstract class used to Run SoftWindows DOS/Windows emulator. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The Windowsemulator class has associations with the following classes:
Class: MgrGui SpawnProgram

7.3.31 xterm Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This is an Abstract class used to represent an UNIX xterm. CM (ClearCase) view is set automatically since a view is already up when AIT Manager is invoked. It is callable from the Unix command line.

Attributes:

None

Operations:

None

Associations:

The xterm class has associations with the following classes:
Class: MgrGui SpawnProgram

7.4 CSCI Dynamic Model

7.4.1 AIT Manager GUI Scenarios

This section gives scenarios for use of the AIT Manager GUI. These scenarios cover various operations which will be invoked from the AITTL Manager GUI. These operations include initiating one of the many COTS tools which will be provided for integration and test activities.

7.4.1.1 Display Main AIT Manager GUI

7.4.1.1.1 Abstract

This scenario occurs whenever the AIT Manager is invoked. Information used to set up the AI&T Manager GUI is gathered from various input files. This information sets the various user preferences invoked for the GUI.

7.4.1.1.2 Stimulus

The initial stimulus is invoked by the Algorithm Integration and Test Operations staff who initiates the AIT Manager program name and options from the Unix command line. Before this operation can occur, the Configuration Management (ClearCase) view must have been set previously.

7.4.1.1.3 Desired Response

Main AIT Manager GUI, including checklist and log, is displayed on user's screen.

7.4.1.1.4 Participating Classes from the Object Model

DpAtMgrCom

DpAtMgrCmdLineData

DpAtMgrInstrConfigData

DpAtMgrChecklistData

DpAtMgrLogData

DpAtMgr

DpAtMgrGuiActivityData

DpAtMgrGui

7.4.1.1.5 Scenario Description

a. User logs into AIT development machine

b. User sets a CM (ClearCase) view

c. User types in AIT Manager executable name and options, if any

d. The default command line options are read from a file; instrument name, instrument config file logical, and static Motif resources file logical.

e. If user typed in any command line options, these options overwrite the defaults, and the new defaults are saved to a file.

f. The instrument configuration data is read from a file: script names, options, menu labels, file logicals, checklist, and log file logicals.

g. The Static Motif resources file is read (all AIT Manager menu data except RUN menu: menu names, locations, program names and options if applicable), and the dynamic Motif resources file is written. This file consists of static motif resources data and instrument config data for RUN menu item.

h. Checklist file data and the last log entry is read.

i. Main AIT Manager GUI is displayed, using dynamic Motif resources file, checklist data and last log entry data.

7.4.1.1.5 Event Trace

Figure 7.4-1 shows the display AI&T main GUI event trace.

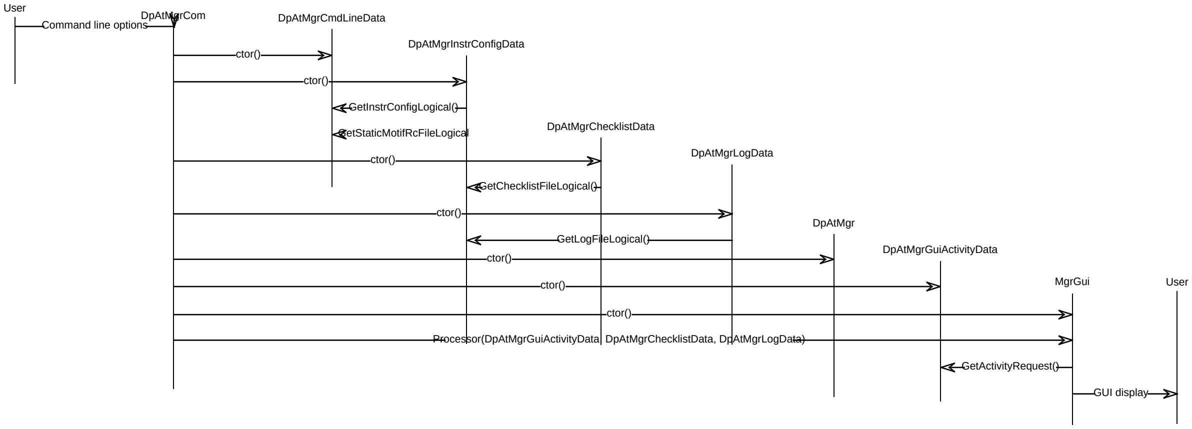


Figure 7.4-1. Display AI&T Main GUI Event Trace Diagram

7.4.1.2 Select a TOOLS Menu Item

7.4.1.2.1 Abstract

This scenario details what happens when the user selects TOOLS menu item *xterm*. The scenario applies equally to the other TOOLS menu items, which are COTS programs (unless noted) useful in the AIT environment:

a. General visualization (IDL)

b. Web browser (Mosaic)

c. Text-graphics viewer (Acrobat)

d. PostScript file viewer (Ghostview)

e. Windows emulator (SoftWindows)

f. Analysis environment (SPARCWorks on AIT Sun Workstation, CODEVision on AIT SGI Workstation)

g. HDF file visualization (EosView, an ECS custom application)

h. Fortran 77 code checker (FORCHECK)

i. Dynamic memory leak detector

7.4.1.2.2 Stimulus

User pulls down TOOLS menu, clicks on *xterm*.

7.4.1.2.3 Desired Response

Unix xterm is displayed. CM (ClearCase) view is automatically set by CM.

7.4.1.2.4 Participating Classes From the Object Model

DpAtMgrGui

7.4.1.2.5 Scenario Description

a. AIT Manager screen previously displayed

b. User pulls down TOOLS menu

c. User clicks on *xterm*

d. Unix xterm is displayed, which automatically has a CM (ClearCase) view set

e. status value is immediately returned indicating whether xterm was successfully spawned

f. User types commands into xterm

f. Other AIT Manager selections are always available during this time

g. xterm remains up until user kills it manually or logs off the machine

7.4.1.2.6 Event Trace

Figure 7.4-2 shows the run tools menu event trace.

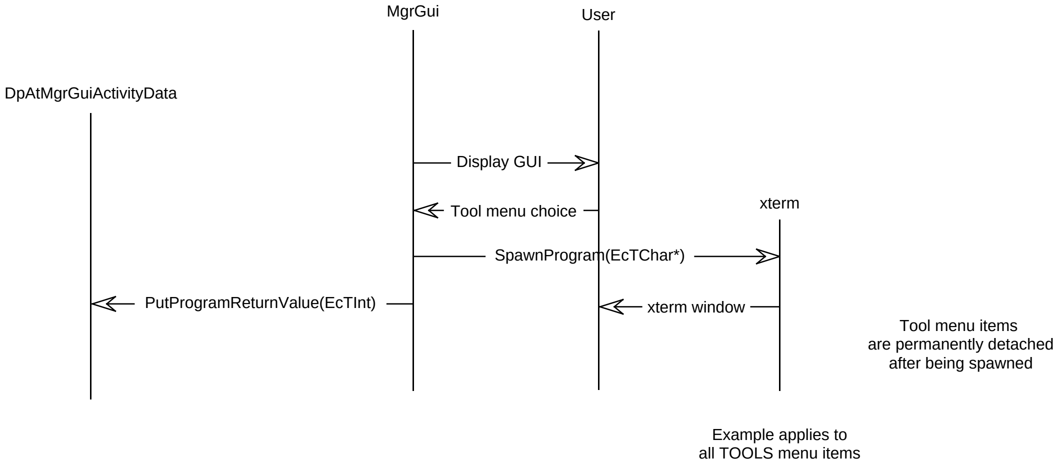


Figure 7.4-2. Run Tools Menu Event Trace Diagram

7.4.1.3 Select a UTILITIES Menu Item

7.4.1.3.1 Abstract

This scenario represents activities that occur when the user selects UTILITIES menu item DpAtMgrCheckHdfFile. It applies equally to other UTILITIES menu items, which are all ECS custom programs:

PGE registration GUI (DpAtPgeRegTool)

CM scripts

DpAtMgrCheckPcfGui

DpAtMgrBinaryFileEnvironmentGui

DpAtMgrCheckProhibFuncGui

This scenario also applies to RUN menu items:

PGE execution GUI (DpAtProcGui)

Instrument-specific scripts

7.4.1.3.2 Stimulus

The Algorithm Integration and Test Operations Staff wants to compare versions of HDF files generated at the SCF to the HDF files generated at the DAAC with SCF Toolkit linked in.

7.4.1.3.3 Desired Response

HDF file checker GUI is displayed

7.4.1.3.4 Participating Classes from the Object Model

DpAtMgrCom

DpAtMgr

DpAtMgrGuiActivityData

DpAtMgrGui

DpAtMgrCheckHdfFil

7.4.1.3.5 Scenario Description

a. AIT Manager screen previously displayed

b. User pulls down UTILITIES menu

c. User clicks on *HDF file checker*

d. HDF file checker GUI is displayed

e. User makes GUI selections, to compare HDF files and/or display HDF metadata

f. Other AIT Manager selections are unavailable during this time

g. User quits from HDF file checker GUI

h. Return value from GUI triggers error handling, if necessary

i. AIT Manager waits for another user selection

7.4.1.3.6 Event Trace

Figure 7.4-3 shows the run utility menu item event trace.

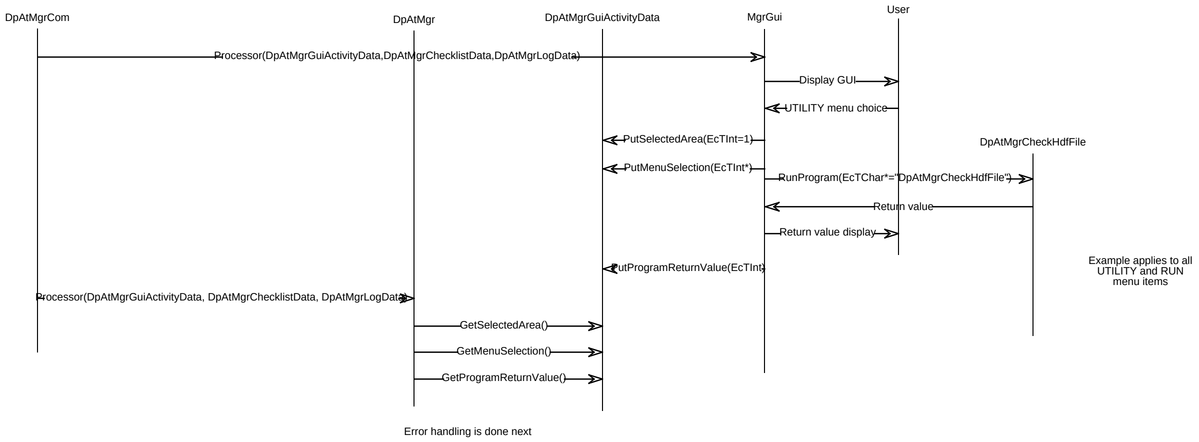

DpAtMgrCom
DpAtMgr
DpAtMgrGuiActivityData
MgrGui
User
Processor(DpAtMgrGuiActivityData,DpAtMgrChecklistData,DpAtMgrLogData)
Display GUI
UTILITY menu choice
PutSelectedArea(EcTInt=1)
DpAtMgrCheckHdfFile
PutMenuSelection(EcTInt*)
RunProgram(EcTChar*="DpAtMgrCheckHdfFile")
Return value
Return value display
PutProgramReturnValue(EcTInt)
Example applies to all UTILITY and RUN menu items
Processor(DpAtMgrGuiActivityData, DpAtMgrChecklistData, DpAtMgrLogData)
GetSelectedArea()
GetMenuSelection()
GetProgramReturnValue()
Error handling is done next

Figure 7.4-3. Run Utility Menu Item Event Trace Diagram

7.4.1.4 Select a Checklist Item

7.4.1.4.1 Abstract

The AIT Manager checklist is a display of items required for check-in, compilation, test execution, etc. of science software. It is displayed as a line of text for each item, with a check box that is checked or unchecked. Each item has a unique ID; the checklist is specific to an instrument configuration. The checklist is manual, in that the user must check it with a mouse click; the program does not check any boxes automatically. Each time a box is checked, the log is updated.

This scenario shows what happens when a box for an item is checked.

7.4.1.4.2 Stimulus

User wants to record that compile and link to SCF Toolkit was successful.

7.4.1.4.3 Desired Response

Box is displayed as checked; log is updated. Item is noted as checked in checklist file after AIT Manager exits.

7.4.1.4.4 Participating Classes from the Object Model

DpAtMgrCom

DpAtMgr

DpAtMgrGuiActivityData

DpAtMgrGui

DpAtMgrChecklistData

DpAtMgrLogData

7.4.1.4.5 Scenario Description

a. AIT Manager has been previously displayed

b. User separately compiles and links science software with SCF Toolkit, e.g., by instrument-specific script or CM script

c. User clicks on check box for "Compile and link with SCF Toolkit" item

d. Checklist item state is changed from unchecked to checked internally

e. Checked box is displayed

f. New log entry is created, displayed on screen, and written to log file

7.4.1.4.6 Event Trace

Figure 7.4-4. shows the select checklist item event trace.

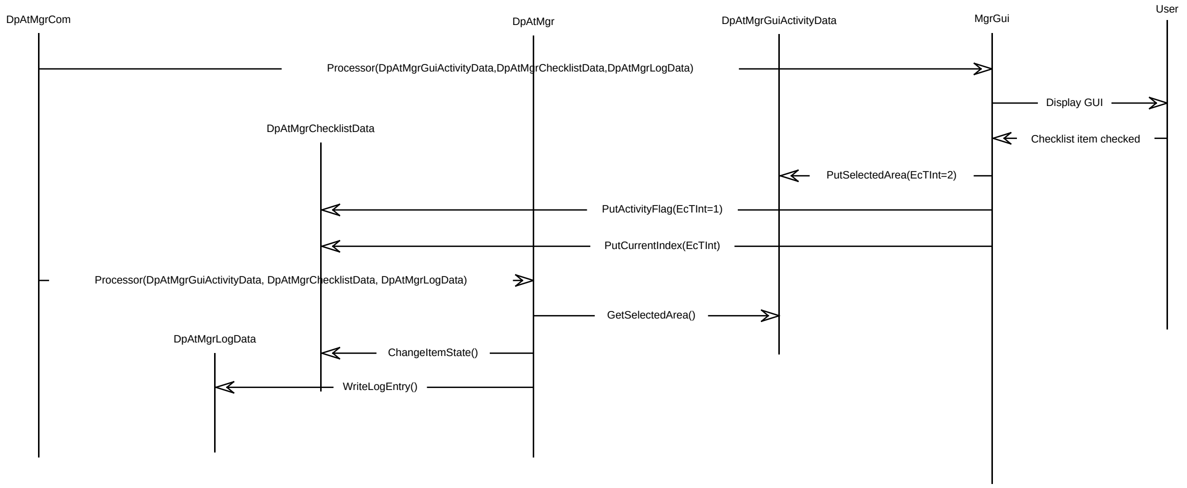

DpAtMgrCom
DpAtMgr
DpAtMgrGuiActivityData
MgrGui
User
Processor(DpAtMgrGuiActivityData,DpAtMgrChecklistData,DpAtMgrLogData)
Display GUI
DpAtMgrChecklistData
Checklist item checked
PutSelectedArea(EcTInt=2)
PutActivityFlag(EcTInt=1)
PutCurrentIndex(EcTInt)
Processor(DpAtMgrGuiActivityData, DpAtMgrChecklistData, DpAtMgrLogData)
GetSelectedArea()
DpAtMgrLogData
ChangeItemState()
WriteLogEntry()

Figure 7.4-4. Select Checklist Item Event Trace Diagram

7.4.1.5 Submit Staging or Destaging Request

7.4.1.5.1 Abstract

These scenarios describe the functions provided to manually initiate the staging or destaging of data, i.e., Science software, Test Data, etc., which are archived in the Data Server archive, The steps taken are very similar to activities performed in the Processing CSCI to automatically initiate staging or destaging by a software application. This will be added to the IR-1 provided functionality for Release A.

7.4.1.5.2 Stimulus

DAAC Operations Staff initiates the staging or destaging of data.

7.4.1.5.3 Desired Response

Data is staged or destaged.

7.4.1.5.4 Participating Classes from the Object Model

DpPrAITManualIF

DsCIESDTReferenceCollector

DsCIRequest

DSCICommand

7.4.1.5.5 Scenario Description

a. AIT Manager has been previously displayed

b. User Requests the staging or destaging of a given data item, using specified reference materials as input guidance.

c. Data Server copies data to specified storage location.

d. Data Server informs AITTL when completed.

7.4.1.5.6 Event Trace

Figure 7.4-5 shows the submit staging request event trace. Figure 7.4-6 shows the submit destaging request event trace.

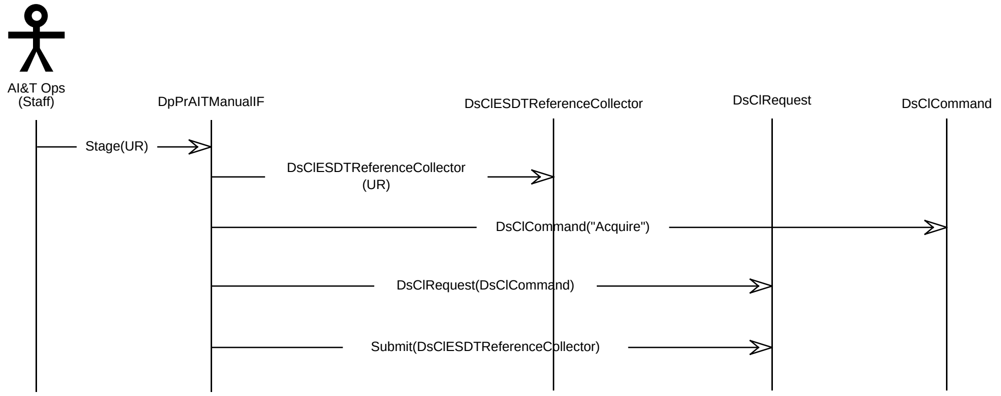


Figure 7.4-5. Submit Staging Request Event Trace Diagram

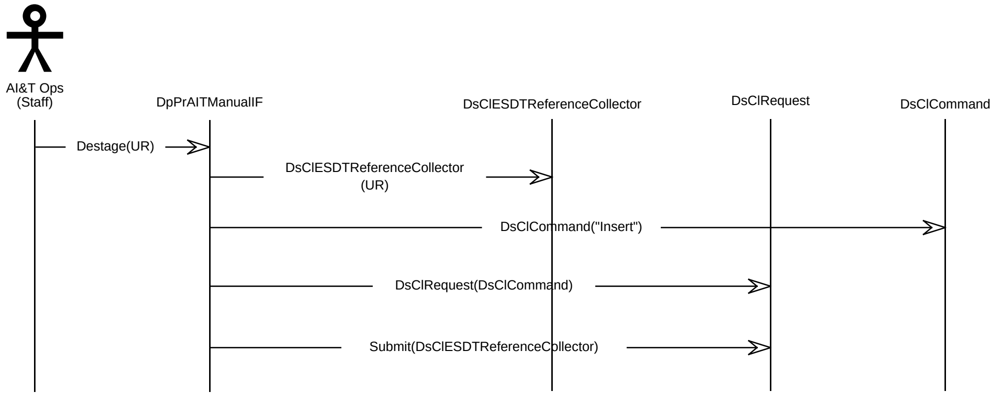


Figure 7.4-6. Submit Destaging Request Event Trace Diagram

7.5 CSCI Functional Model

The sections which follow present a context diagram for each of the components of AITTL. These components, or tools, correspond roughly to the requirements mappings found in 304-CD-002-001, Science and Data Processing Segment (SDPS) Requirements Specification for the ECS Project. They also correspond to the computer software components (CSCs) listed below in Section 7.5, AITTL Structure.

7.5.1 Viewing Science Software Documentation

The context diagram for the tools to display and/or print science software documentation is shown in Figure 7.5-1.

The operator issues a command *(Document Viewing Command)* to view a particular document on the display *(Displayed Document)*, or to print a hard copy *(Printed Document)* of a document. Softcopy of the document *(Document)* is stored online in the local integration and test area *(Science Software)*.

7.5.2 Checking Coding Standards

The context diagram for the standards checkers is shown in Figure 7.5-2.

The operator issues a command *(Standards Checking Command)* to check for compliance of a particular script *(Shell Script)* or source file *(Source Code)*, or set of files, with certain standards and/or guidelines. The operator also must supply the standards checkers with the required standards *(Standards)* and guidelines *(Guidelines)* (only once, when the tools are configured). The results of the check *(Standards Checking Results)* are displayed on the console. Reports may be generated as well, which may be displayed *(Displayed Standards Checking Reports)*, printed *(Printed Standards Checking Reports)*, and saved as softcopy *(Standards Checking Report)*.

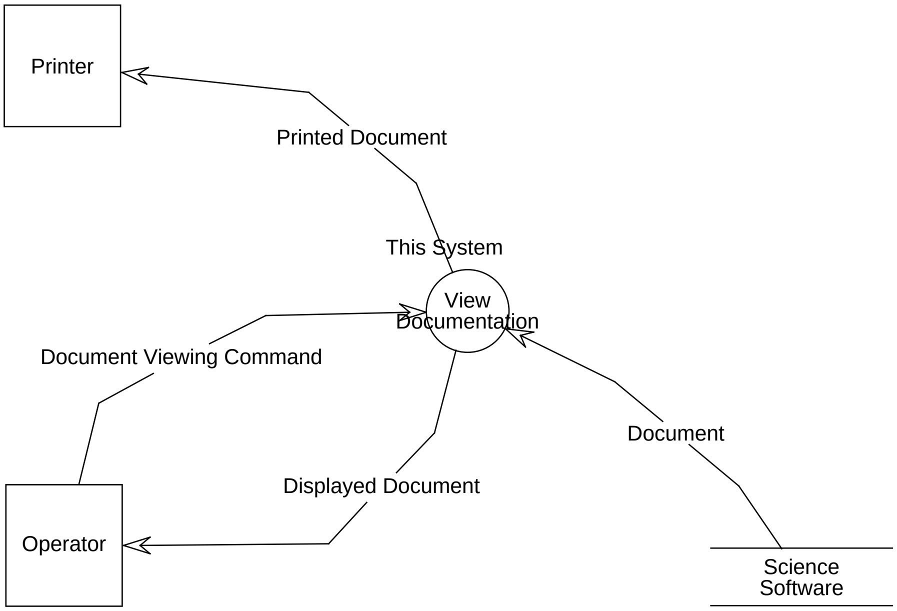


Figure 7.5-1. Data Flow Diagram: View Documentation

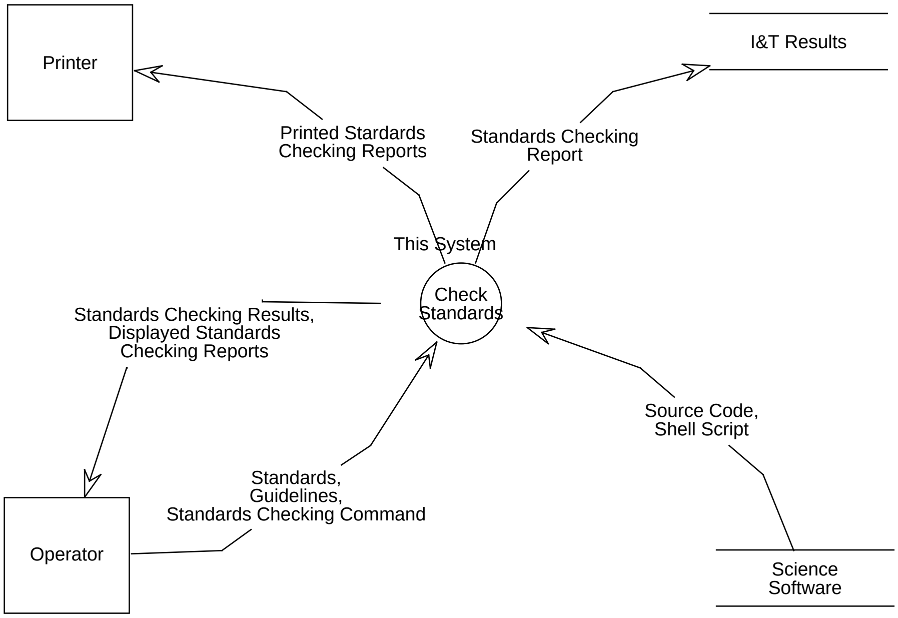

Printer
I&T Results
Printed Stardards Checking Reports
Standards Checking Report
This System
Check Standards
Standards Checking Results, Displayed Standards Checking Reports
Source Code, Shell Script
Standards, Guidelines, Standards Checking Command
Operator
Science Software

Figure 7.5-2. Data Flow Diagram: Check Standard

7.5.3 Analyzing the Code

The context diagram for the static and dynamic code checkers is shown in Figure 7.5-3.

The operator issues a command *(Code Analysis Command)* to make certain checks, either statically on the source files *(Source Code)*, or dynamically on the executables *(Executable)*. The results

(Code Analysis Results) of the checks are displayed on the console. Reports may also be generated, and these may be displayed *(Displayed Code Analysis Report)*, printed *(Printed Code Analysis Report)*, or saved as softcopy *(Code Analysis Report)*.

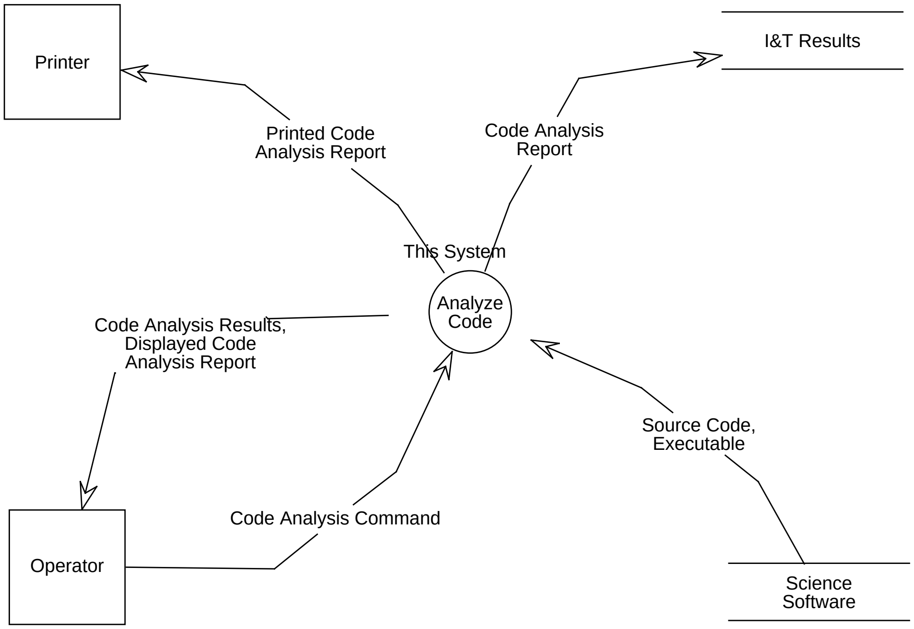


Figure 7.5-3. Data Flow Diagram: Analyze Code

7.5.4 Examining the Data

The context diagram for the data visualization tools is shown in Figure 7-5-4.

The operator issues a command *(Data Visualization Command)* to examine a particular data file. The data visualization tools may display the data in the form of a data dump *(Displayed Data Dump)*, a two- or three-dimensional plot *(Displayed Plot)*, or an image *(Displayed Image)*. These displays may also be printed *(Printed Data Dump, Printed Plot, Printed Image)* or saved as softcopy *(Data Dump, Plot, Image)*.

7.5.5 Comparing Data Files

The context diagram for the file comparison utility is shown in Figure 7.5-5.

The operator issues a command *(File Comparison Command)* to find all differences between two data files *(Data Files to Compare)*. Normally, one of these files will have been generated by running a particular test case at the SCF, while the other will have been generated by running the same test case at the DAAC. If there are precision differences between the SCF and DAAC processing platforms, there will be corresponding differences between the two data files which should be ignored. Therefore, the operator will also supply a threshold *(File Comparison Threshold)* for masking out these types of differences. The results *(File Comparison Results)* of the file comparison are displayed on the console. Reports may also be generated, and these may be displayed *(Displayed File Comparison Report)*, printed *(Printed File Comparison Report)*, or saved as softcopy *(File Comparison Report)*. Please note that this section applies directly to HDF files. For binary files, the ECS supplies a programming environment which the DAAC Operator used to write custom binary file comparison code.

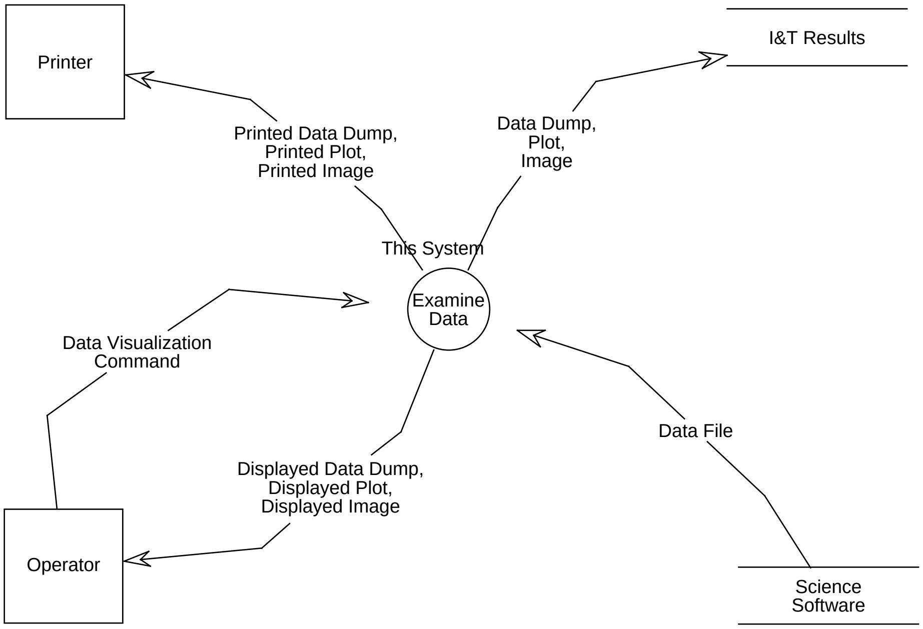


Figure 7.5-4. Data Flow Diagram: Examine Data

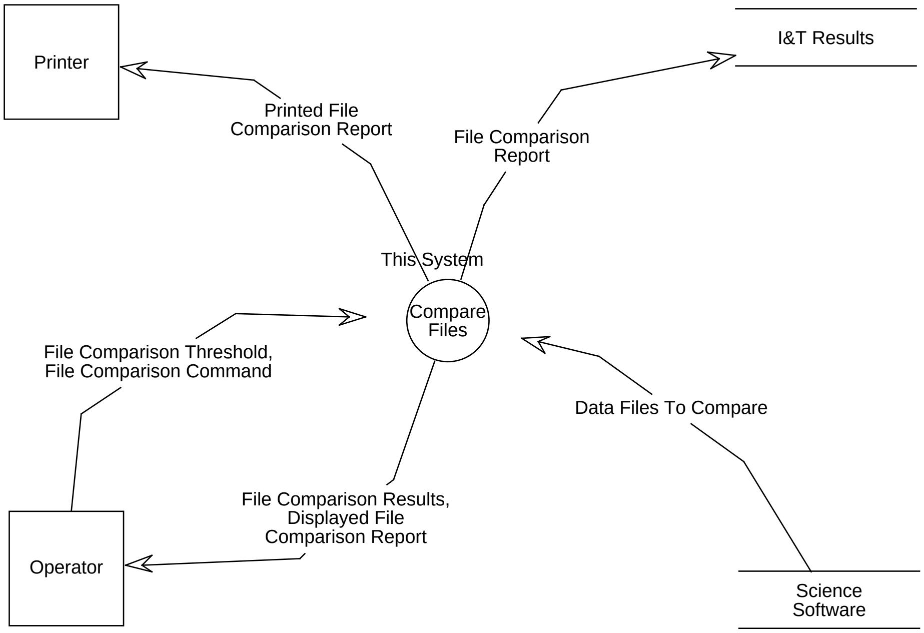


Figure 7.5-5. Data Flow Diagram: Compare Files

7.5.6 Measuring Resource Requirements

The context diagram for the profiling tools is shown in Figure 7.5-6.

The operator issues a command *(Profiling Command)* to measure certain resource usage statistics, such as CPU time, memory usage, I/O accesses, disk space requirements, etc., for a specified process or procedure *(Shell Script, Executable)*. The profiling results *(Profiling Results)* are displayed on the console. Reports may also be generated, and these may be displayed *(Displayed Profiling Report)*, printed *(Printed Profiling Report)*, or saved as softcopy *(Profiling Report)*.

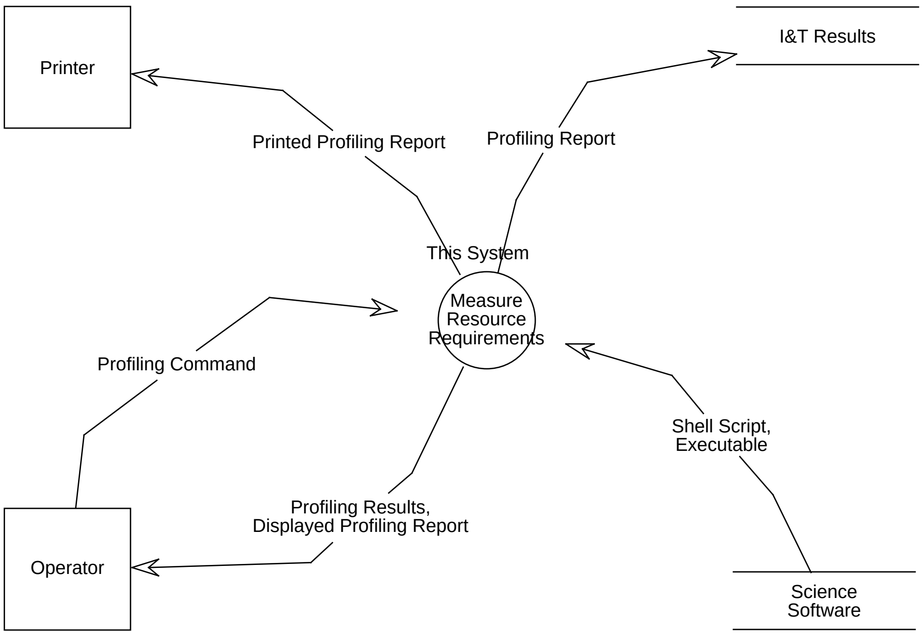


Figure 7.5-6. Data Flow Diagram: Measure Resource Requirements

7.5.7 Updating the Data Server

The context diagram for the GUI for archiving the science software files on the data server is shown in Figure 7.5-7.

The purpose of this tool is to allow the user to add a new or updated set of science software files to the data server once the software has passed acceptance testing. This set of files will probably include not only the delivered files *(Science Software)*, but also some of the reports *(I&T Report)* generated during integration and test. The operator must also supply metadata *(Science Software Metadata)* for the science software. The operator may issue various commands *(Data Server Update Command)* to the update tool, such as commands to view the current contents of the data server, commands to add the new science software files to the data server, commands to update the metadata, and so on.

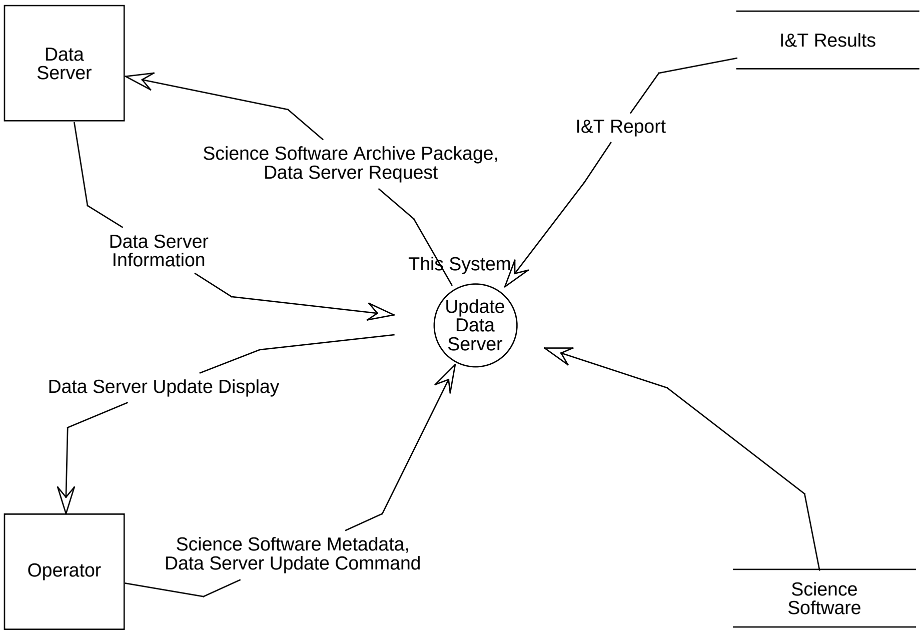

Data Server
I&T Results
I&T Report
Science Software Archive Package, Data Server Request
Data Server Information
This System
Update Data Server
Data Server Update Display
Operator
Science Software Metadata, Data Server Update Command
Science Software

Figure 7.5-7. Data Flow Diagram: Update Data Server

Requests *(Data Server Request)* are sent to the data server to get information about its contents; the data server responds with the requested information *(Data Server Information),* which is then displayed *(Data Server Update Display)* on the operator's console. Requests *(Data Server Request)* to add a new set of science software files are also sent to the data server, along with the set of files *(Science Software Archive Package)* to be archived; the data server responds with the status of the request *(Data Server Information),* which is also displayed on the console *(Data Server Update Display).*

7.5.8 Updating the PGE Database

The context diagram for the GUI for updating the PGE database is shown in Figure 7.5-8.

Information to be loaded into the PGE database is taken from the profiling reports *(Profiling Report)* (see Section 7.3.6, Measuring Resource Requirements). The operator issues commands *(PGE Database Update Command)* to add the information to the database (or to view the contents of the database, or add, delete, or modify entries). The PGE Database is part of an overall Database referred to in the Planning and Processing CSCI design as the PDPS Database. The PDPS Database is a shared database used by Planning, Processing and Algorithm Integration and Test to manage their persistent data. These commands *(PGE Database Request)* are passed onto the PGE Database by the GUI, along with the new or modified database entries *(PGE Installation Package)*. Information or status *(PGE Database Information, PGE Installation Package)* is returned by PGE Database, and this information *(PGE Database Update Display)* is displayed on the operator's console.

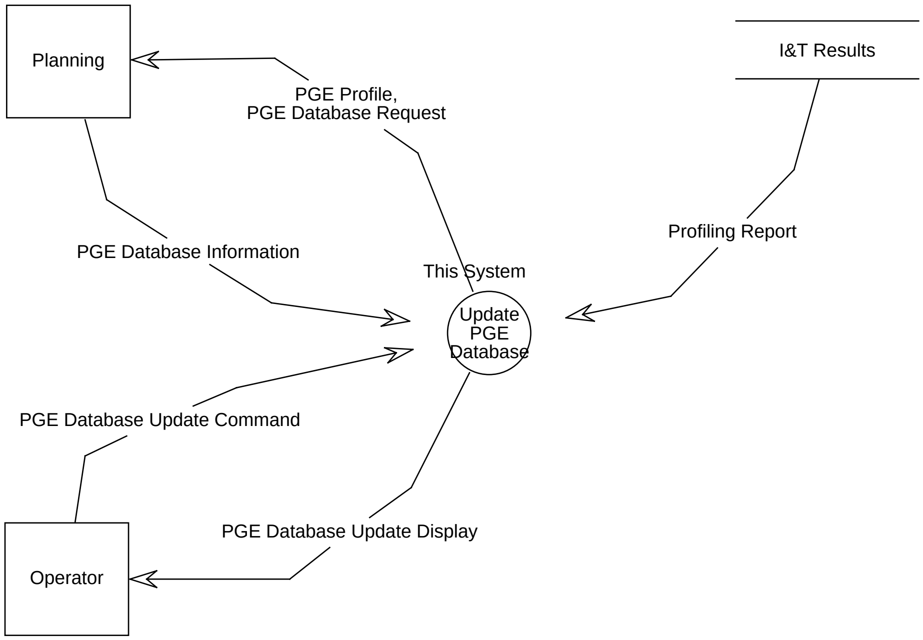


Figure 7.5-8. Data Flow Diagram: Update PGE Database

7.5.9 Writing Reports and Maintaining Logs

The context diagram for tools to generate and maintain the integration and test reports and logs is shown in Figure 7.5-9.

The operator issues commands *(Report Management Command)* to create/modify specified reports *(I&T Report)* and logs *(I&T Log)*, supplying the required information *(Report Information, Log Information)* and/or using information from other manually- or tool-generated reports *(I&T Report)*, and using pre-generated report templates *(I&T Templates)*. The operator may also display *(Displayed I&T Report, Displayed I&T Log)* or print *(Printed I&T Report)* any report or log, and any authorized user may display the log.

7.5.10 Manually Staging Inputs

The context diagram for the tool for manually staging inputs is shown in Figure 7.5-10.

The operator issues commands *(Data Staging Command)* to manually stage a specified data file from the data server. The request *(Data Transfer Request)* is passed onto the data server, which returns the requested file *(Data File)*, which is then placed in the staging area *(Staged Data)*. Status *(Data Staging Status)* is displayed on the operator's console.

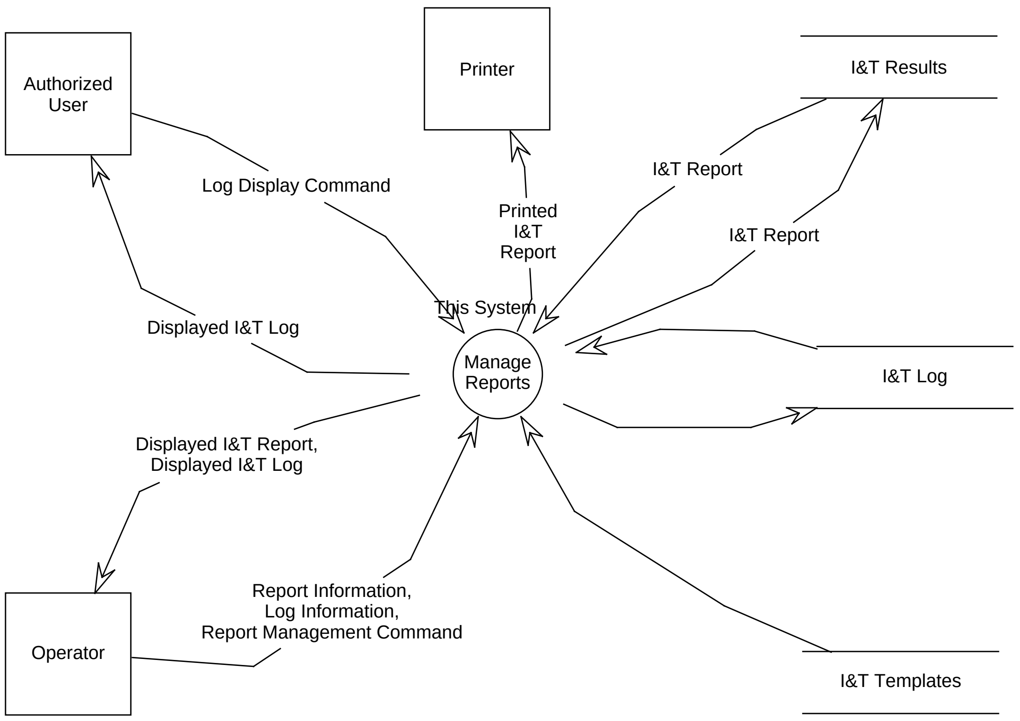


Figure 7.5-9. Data Flow Diagram: Manage Reports

7.5.11 Displaying Product Metadata

The context diagram for the tool for displaying product metadata is shown in Figure 7.5-11.

The operator issues commands *(Metadata Command)* to display the product metadata *(Displayed Product Metadata)* for a specified data file *(Data File)*. The metadata may also be extracted into a report, which may be displayed *(Displayed Product Metadata Report),* printed *(Printed Product Metadata Report)*, or saved as softcopy *(Product Metadata Report)*.

7.6 AITTL Operational Scenarios

The following three operational scenarios are intended to give an idea of how the science software integration and test process will be done, as well as illustrating where in the process the AITTL tools might be used by the SCF and DAAC I&T personnel. These scenarios were developed by the ECS Science Office and assume ECS Release A.

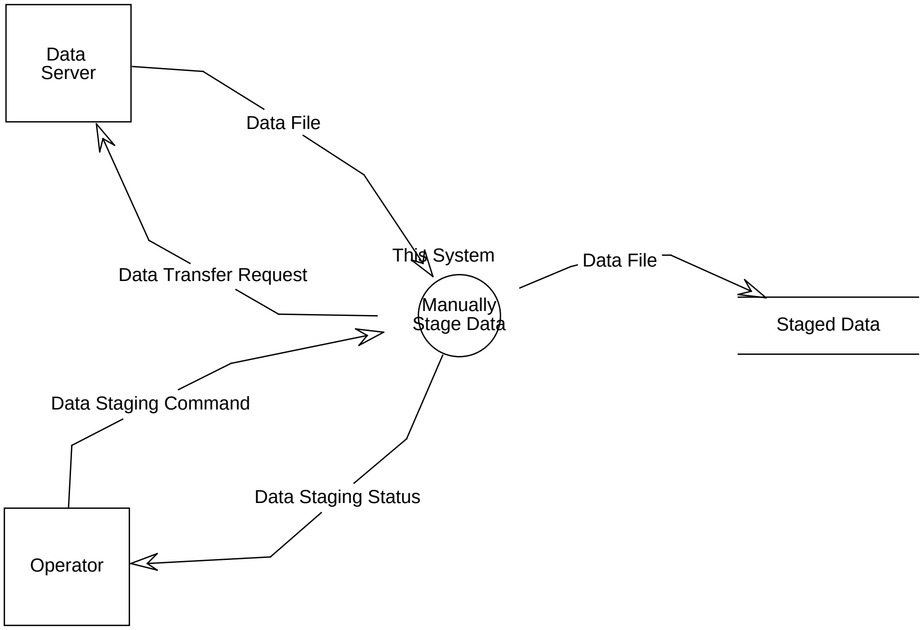


Figure 7.5-10. Data Flow Diagram: Manually Stage Data

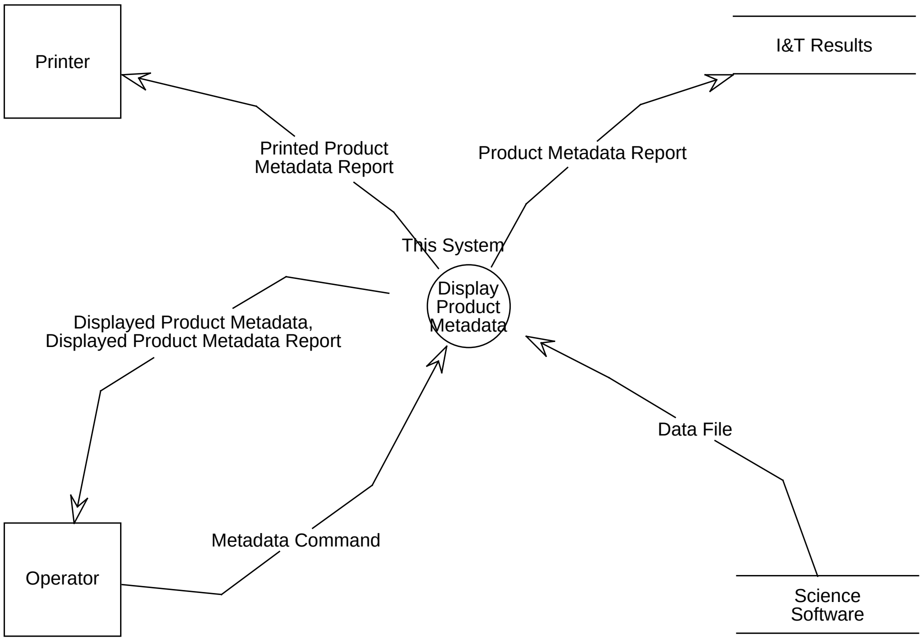


Figure 7.5-11. Data Flow Diagram: Display Product Metadata

7.6.1 Engineering Version for AM-1

1) SCF sends 200 8-mm tapes containing test data files, expected test result files and associated metadata file. These tapes contain 11,000 files averaging 50 MB each.

2) DAAC Ingest - Distribution Technician mounts the delivered tapes and transfers these files to the Data Server/Archive.

3) DAAC Operations Supervisor has an Investigator Account opened for the SCF, an Investigator Directory created, and a CM storage pool allocated specifically for use in SSI&T.

4) The SCF transfers the science software (e.g., via ftp) to the Investigator Directory, and checks the source, includes, makefiles, scripts and libraries into the CM storage pool.

5) DAAC Resource Planner reserves 1 CPU of a Processing Server during day shift for use by SSI&T for the requested duration.

6) SCF remotely (e.g., via telnet) performs stand-alone testing of individual PGEs and PGE chains, employing the various AITTL tools in a "batch" mode (e.g., the data visualization tool is used to generate a graphics file, which is the transferred to the local site for display). Test data files are requested as needed from the Data Server/Archive.

7) When standalone testing is complete, SCF labels the "Delivery" elements in the SSI&T CM storage pool.

8) DAAC CM Administrator builds binary executables, checks out PGE scripts and submits these to Data Server (along with appropriate metadata).

9) DAAC Production Planner/Scheduler enters the test PGE information (e.g., PGE ID and Version No., resource profiles, inputs, activation rules) into the Planning Data Base.

10) The SCF subscribes to the test PGE outputs.

11) DAAC System Tester requests test data from the Data Server and stages it to Ingest.

12) Processing is automatically performed, and the PGE output and processing log files are automatically sent to the SCF.

13) SCF verifies PGE output.

14) The SCF checks the "Delivery" items out from the CM storage pool and transfers these elements (employing a DCE client) from the Investigator Directory to the SCF.

15) The DAAC performs cleanup work.

The CM Administrator deletes the SSI&T storage pool. The DAAC Production Planner removes the SSI&T PGEs from the planning Data Base and cancels the Production Request(s). The DAAC closes the Investigator Account and removes the Investigator Directory.

7.6.2 Launch-Ready Version for TRMM

1) SCF requests that 11,000 previously delivered test data files be staged from deep archive to Data Server/Archive on-line storage.

2) The Data Server/Archive retrieves these files from deep archive.

3) DAAC Operations Supervisor has a User Account opened for the SCF, an Investigator Directory created, and a CM storage pool allocated specifically for use in SSI&T.

4)-14) Steps 4-14 of the “Engineering Version for AM-1” scenario are performed.

15) The DAAC CM Administrator copies the “Delivery” elements from the SSI&T CM storage pool to a Master CM storage pool. Then deletes the SSI&T CM storage pool.

16) The Production Planner/Scheduler enters Production Requests for product generation using the Launch-ready PGEs.

(The DAAC and SCF complete the operations readiness activities, and the new PGE is promoted to production status.)

7.6.3 Science Software Upgrade

1) SCF requests that 24 previously delivered test data files be staged from deep archive to Data Server/Archive on-line storage.

2) The Data Server/Archive retrieves these files from deep archive.

3) DAAC Operations Supervisor has a User Account opened for the SCF, an Investigator Directory created, and a CM storage pool allocated specifically for use in SSI&T.

4)-14) Steps 4-14 of the “Engineering Version for AM-1” scenario are performed.

15) The DAAC CM Administrator checks-in to the Master CM storage pool all of the elements from the SSI&T CM storage pool which are labelled “Delivery.” The SSI&T CM storage pool is deleted.

7.7 AITTL Structure

Table 7-1 provides a list of the Computer Software Components in the AITTL CSCI.

Note that all software is callable from the Unix command line; alternatively, it may be called from the AIT Manager GUI. Command line versions are shown in brackets [], where they differ from the GUI versions.

In implementing custom code, some SDP Toolkit functions are reused, primarily to track AIT configuration files and manage temporary files.

Table 7-1. AITTL Computer Software Components (1 of 2)

CSC	Description	Implementation
Documentation Viewing Tools	Tools for displaying and/or printing the science software documentation	SoftWindows/MS Office Ghostview
Standards Checkers	Tools for checking if science software follows prescribed coding standards.	Native compilers FORCHECK
Code Analysis Tools	Tools for checking for code memory leaks, etc.	CASEVision SPARCWorks
Data Visualization Tools	Diagnostic tools which display input, output, and intermediate data files as data dumps, plots, and/or images.	IDL

Table 7-1. AITTL Computer Software Components (2 of 2)

CSC	Description	Implementation
ECS HDF Visualization Tools	Provides the capability to view any ECS-HDF formatted files.	EOSView
HDF File Comparison Utility	Tool for finding differences between two HDF files,	DpAtMgrCheckHdfFile
Binary File Comparison Environment	Tool for assisting DAAC user in writing custom code to find differences between two binary files	DpAtMgrBinaryFileEnvironmentGui
Profiling Tools	Tools for measuring the resource requirements of the science software	CASEVision
PGE Processing GUI	GUI for executing science software	DpAtProcGui
Update Data Server GUI	GUI used for staging or destaging data.	DpPrAITManualIF
Update PGE Database GUI	GUI for registering a PGE with the processing system	DpAtPgeRegTool
Report Generation Tools	Tools for writing miscellaneous reports and for maintaining the integration and test log	SoftWindows/MS Office DpAtMgrCom
SDP Toolkit-related Tools	Tool to check Process Control File format; tool to check that no prohibited functions are used	DpAtMgrCheckPcfGui [pccheck.sh] DpAtMgrCheckProhibFunc [DpAtMgrCheckProhibFuncCom]
Product Metadata Display Tool	Tool for displaying the product metadata	DpAtMgrCheckHdfFile

7.7.1 Documentation Viewing Tools

This CSC contains the tools that the integration and test personnel will use to view the science software documentation. The tools only need to be able to display a document on a console and print the document—there is no editing capability required. The list of formats that will be accepted is given in 304-CD-002-001, Science and Data Processing Segment (SDPS) Requirements Specification for the ECS Project (see requirements S-DPS-40100 and 40110).

The Microsoft Office automation tools, in conjunction with the SoftWindows Windows emulator for Sun, is used here.

7.7.2 Standards Checkers

This CSC contains the standards checking tools. Native language compilers satisfy the requirements here, except in the case of FORTRAN 77. For that language, a COTS standards checker is used, because most if not all compilers support a near-standard subset of ANSI FORTRAN 77; it is expected that essentially all FORTRAN 77 science software will contain these extensions to the ANSI standard.

7.7.3 Code Analysis Tools

These tools are for enabling DAAC personnel to determine the causes of such problems as memory leaks. They include the powerful analysis environments SPARCWorks (on the Sun) and CASEVision (on the SGI).

7.7.4 Data Visualization Tools

This CSC contains the data visualization tool IDL, required by the integration and test personnel to examine input, output, and intermediate data files for diagnostic purposes (but not for QA).

7.7.5 ECS HDF Visualization Tools

This CSC is the ECS-developed EOSView tool for viewing ECS HDF files. The tool is reused from the WKBCH CSCI.

7.7.6 HDF File Comparison Utility

This CSC contains the HDF file comparison utility. It is implemented by a custom IDL program, which includes a GUI front end. Difference data may optionally be displayed as text, as a line graph with tolerances, or printed.

The first time a given HDF standard product is compared, the DAAC user must manually input data tolerances that s/he reads by eye from a text file delivered with the science software.

7.7.7 Binary File Comparison Environment

This CSC contains the binary file comparison environment. This is implemented by a GUI from which the user can choose example code and function utilities to cut and paste for making a custom file differencing tool, tailored to the particular binary file format. This format is also available from the GUI, as delivered in a text file along with the science software.

7.7.8 Profiling Tools

This CSC contains the profiling (i.e., resource requirement measurement) tools. These tools must satisfy any of the AITTL profiling requirements that are not satisfied by the development environments or operating systems supplied by the AITHW CI. CASEVision covers this.

7.7.9 PGE Processing GUI

This CSC contains the GUI used to execute a PGE, which is called DpAtProcGui.

7.7.10 Update PGE Database GUI

This CSC contains the GUI used to register a PGE in the processing system, which is called DpAtPgeRegTooli.

7.7.11 Report Generation Tools

This CSC contains the tools to write and maintain the integration and test reports and logs. All but the log requirements are satisfied by SoftWindows/MS Office; the AIT Manager covers the need for a log.

7.7.12 SDP Toolkit-related Tools

These tools (a) check the format of a delivered Process Control File and (b) determine whether any prohibited functions are used, which might interfere with the processing system. Modules DpAtMgrCheckPcfGui (which is a wrapper on an existing SDP Toolkit utility) and DpAtMgrCheckProhibFuncGui are used, respectively.

7.7.13 Product Metadata Display Tool

This CSC contains the tool for displaying product metadata. It is implemented by DpAtMgrCheckHdfFile, the same tool as in "HDF File Comparison Utility" above.

7.8 CSCI Management and Operation

7.8.1 System Management Strategy

This section discuss the management and operation of the AITTL CSCI. It addresses how the CI is managed at the local level and supports system level management and operations.

Three key concepts define the management and operations of this CSCI. These are that:

a. The AIT process is essentially standalone relative to the rest of the ECS system

b. AIT is operations and operational procedure driven.

c. The AITTL CSCI must support the flexibility of operations required of an I&T activity.

The following paragraphs discuss these features further, indicating how the AITTL CSCI supports this view of management and operations.

Independent Operations

From the point of view of system management and operations, the AITTL operates in essentially a standalone mode, apart from the rest of the ECS. Its purpose is to support the integration and test of Science Software with the ECS system at the DAACs. This is a non-operational CSCI in the sense that it is not involved with the handling or processing of science telemetry from EOS supported spacecraft or instruments. During the operational phases, hardware elements on which the AITTL will run during integration and test (I&T) will be specially configured so that they do not interfere with operations, and operations does not interfere with the test process.

The AITTL does not participate with the system management services provided through MSS for operational enterprise management, such as fault management and startup & shutdown. Note however, that the hardware platforms on which the AITTL tool set is used during I&T will be monitored via MSS agents for hardware faults. The tool set makes use of MSS provided configuration management tools to place the science software under configuration management during the I&T process for eventual migration into operational configuration control databases. The operational procedures AITTL call for the use of ECS standard'trouble ticketing' utilities for recording of problems identified during I&T. Other system services, such as ingest and data server access, will be used to establish the environment for the I&T process. But the I&T process and the AITTL CSCI will be decoupled from the operational ECS system.

Operations Procedure Driven

A key concept for the operation involving AITTL is that the AIT process, as with almost any integration and test activity, is a people-intensive activity. It is driven by the operational procedures established by the DAAC-AIT team in consultation with the science software developers. The management of the AITTL is dictated by the management of AIT operations procedures. The tool set supports this view in that it is a collection of tools that is called upon, one at a time, to address one of the steps in the sequence of activities in the procedure. Evolution of the I&T operations procedures (as the result of increased understanding of the process and the software) is supported because there is no underlying assumption within the tool set implementation with regard to the details of the I&T operations procedures.

Flexibility of Operations

A critical feature of the AITTL CSCI is that it must exhibit great flexibility to support the dynamic environment that is a part of the I&T process. The AIT process, particularly during the TRMM Release period, will be greatly constrained in the time allotment. The personnel (operations, ECS, and science software) involved in the process must be able to adapt quickly to address unexpected situations as they arise during I&T. The tool set is flexible enough to support the many variations that can be expected to occur during the many science software integration and test events. Again, because the tool set does not assume a particular operations procedure, the tools can be readily adapted to variations in the procedures, from one instrument team to another. The tool set can support adaptation of the procedures'on-the-fly' as problems are detected and alternate approaches to the I&T process are investigated. This is because the AITTL CSCI is a collection of tools that support the I&T operations procedure—as the operations procedure is updated or modified, the tool set support can be redirected.

7.8.2 Operator Interfaces

This subsection describes the operator user interfaces provided by the AITTL CSCI to DAAC operations personnel. A general description of the framework and methodology employed for the development of these interfaces can be found in Section 4.5. of the Detailed Design Overview (305-CD-001-001). This subsection augments that information with additional design information which is specific to the Algorithm Integration & Test CSCI.

The operator user interfaces for the algorithm integration & test environment are custom and COTS provided interfaces. This custom graphical interface will be created with the aid of the Integrated Computer Solutions' Builder Xcessory. Builder Xcessory enables the developer to manage Motif graphical user interface projects by providing a WYSIWYG, drag and drop, visual development environment. Once an interface is constructed, Builder Xcessory will generate C++ code which represents the GUI and encapsulates the C-based Motif Widget set. The generated C++ code can then be combined with other Processing CSCI specific code.

7.8.2.1 Off-The-Shelf Interfaces

The Job Scheduling COTS products, AutoSys and AutoXpert, provide GUIs to interface with their applications. The AutoSys GUI, known as the Operator Console, provides capabilities to manage and monitor a schedule of jobs. The GUI provides visibility to the job stream as well as supporting alarm and monitoring mechanisms. Also provided is a set of job interfaces which allow the operator to interact with a job. These interfaces support job creation, modification, cancellation, suspension, and modification. More information on the AutoSys product can be found in Section 4.1.4 and the underlying subsections.

7.8.2.2 Algorithm Integration & Test CSCI User Interfaces

This section is intended to describe the data that may be displayed for the operations of the AITTL CSCI's applications. The exact definition of the GUI will be decided by requesting user suggestions and through demonstrating prototypes.

AI&T Manager GUI

The AI&T Manager GUI is an interface which provides the operations staff with the capability to perform various AI&T activities which are defined as data visualization, updating quality assurance metadata, and subscribing to data. The AI&T Manager GUI will be constructed with custom code which interfaces to the PDPS Database for storage of PGE Profile and other AI&T persistent data.

As part of the GUI, utilities will be provided to initiate data visualization tools, such as EOSVIEW, which will be used to visualize a science data product. Other tools needed to perform AI&T duties, including standards checkers, memory analyzers, and office automation tools can also be invoked from this tool. Also, provided is an interface to manually initiate the staging of data to or destaging of data from the Science Data Server CSCI. Most of the GUI interfaces being developed for the AITTL CSCI are being developed as prototypes for IR-1. Information gathered through the use of these initial prototypes will be fed into the development of the Release A AITTL CSCI GUI.

7.8.3 Reports

A variety of ad-hoc and canned reports will be available to the DAAC operations staff to assist in the monitoring of the activities associated with the Algorithm Integration & Test CSCI. These reports are readily accessible given that the Algorithm Integration & Test CSCI persistent data is maintained in the PDPS Database, a SYBASE RDBMS. Also, ECS application management information is maintained in the MSS database, which is used to log system events. The canned reports will include the following:

a. PGE Profile Reports—these reports will be used to catalog the resource profile information associated with a PGE. This information, which includes generation size of PGE Output data, CPU Wall Clock Time Used, CPU actual time Used, I/O Operations, etc., is captured over a series of PGE executions. A profile will be captured for each type of machine, i.e., Sun, SGI, etc., for which the PGE is to execute. Statistics will be collected to establish standard deviations, variances, and averages of resource profile values. These reports will be used to collate this information for a PGE, for a type of resource, or for a given group of PGEs used to fulfill a Production Request.

b. I & T Activity Report—these reports will capture information about the activities which have occurred and activities which are occurring in the Algorithm Integration and Test environment.

c. PGE Database Update Report—these reports will capture information to track the updates which have occurred in the PGE Database.

d. PGE I&T Reports—these reports capture information on PGEs as they progress through the AI&T process. These reports will be used to trouble shoot problems and will provide tracking and trend analysis guidance.

   1. Code Analysis Report

   3. Standards Checker Report

   3. File Comparison Reports

   4. AI & T Discrepancy Reports

5. Inspection Reports
6. Integration Reports
7. Acceptance Reports

e. Algorithm Integration & Test Management Reports—These reports will provide the operations staff information on Algorithm Integration and Test application software events which have occurred. This information will be available from the MSS database.

f. Job Status/Event Reports—These reports will provide information on the history of the AutoSys job schedule. All status and event changes for a job will be logged in the AutoSys Database. Any information associated with a job can reported on.

Other ad-hoc reports can be defined to assist the Algorithm Integration & Test Operations staff in performing their activities. The PDPS Database is the repository used to maintain information on Production Requests and associated Data Processing Requests, Data Subscriptions, PGE Profiles, etc. These reports can be used to track modifications and provide historical information on these data objects. Because of the used of a consistent RDBMS throughout ECS, the sharing of information between different databases is simplified and will allow for consistent definitions for any number of reports.

This page intentionally left blank.

8. SPRHW - Science Processing HWCI

The Data Processing Subsystem (DPS) consists of three (3) hardware CIs: (1) Science Processing, (2) Algorithm Integration and Testing (AI&T), and (3) Quality Assessment and Monitoring (AQAHW). DPS is responsible for managing, queuing, and executing processes on a specified set of processing resources at each DAAC site.

The Science Processing HWCI (SPRHW) is the primary HWCI in the Processing Subsystem and contains staging (working storage), input/output (I/O), and processing resources necessary to perform routine processing, subsequent reprocessing, and Algorithm Integration & Test (AI&T). SPRHW CI consists of two components: (1) Science Processing, and (2) Processing Queue Management. The Science Processing component is broken up into processing "clusters" that are chains of processing, I/O and staging resources configured to deal with unique processing requirements to which they are designed, tuned and allocated. This does not imply that the only use of a processing cluster, or a specific compute server on that cluster, is limited to one specific class of instrument algorithms alone. The AI&T cluster is also a test and backup cluster and provides a "fail soft" environment for production processing.

The Data Processing Subsystem, in conjunction with the Planning Subsystem, plans for and allocates resources to any task suited to the inventory of available resources Figure 8-1 depicts a logical topology of the Data Processing Subsystem and Planning Subsystem showing their components and major interconnections.

8.1 HW Design Drivers

For Release A sizing, Science Processing platform class(es) recommendations for ECS are primarily based upon the CDR Technical Baseline containing Ad Hoc Working Group on Production (AHWGP) capacities (January 1995), scalability of hardware, evolvability of hardware and software, and cost/performance. Processor platform class(es) selection will be readdressed on an as needed basis to meet Release A CDR and Release B goals at a minimum.

The overall hardware design for ECS is that of a heterogeneous computing environment. Recommendations at this time are specifically for IR-1 and Release A, with a "Look Ahead" to Release B for scalability and evolvability. The candidate hardware is tailored to DAAC unique instrument processing needs.

Hardware recommendations are based upon: (1) Cost/Performance tradeoffs, (2) Analysis of AHWGP data incorporating ESDIS phasing and efficiency factors, (3) ECS Science and Technology Lab Prototyping and (4) Trade Studies (i.e., Distributed and Parallel Processing, Production Platform Families, Production Topologies). The hardware selection and the design will support a phased procurement, heterogeneous architectures, use of heritage software, and multivendor platforms.

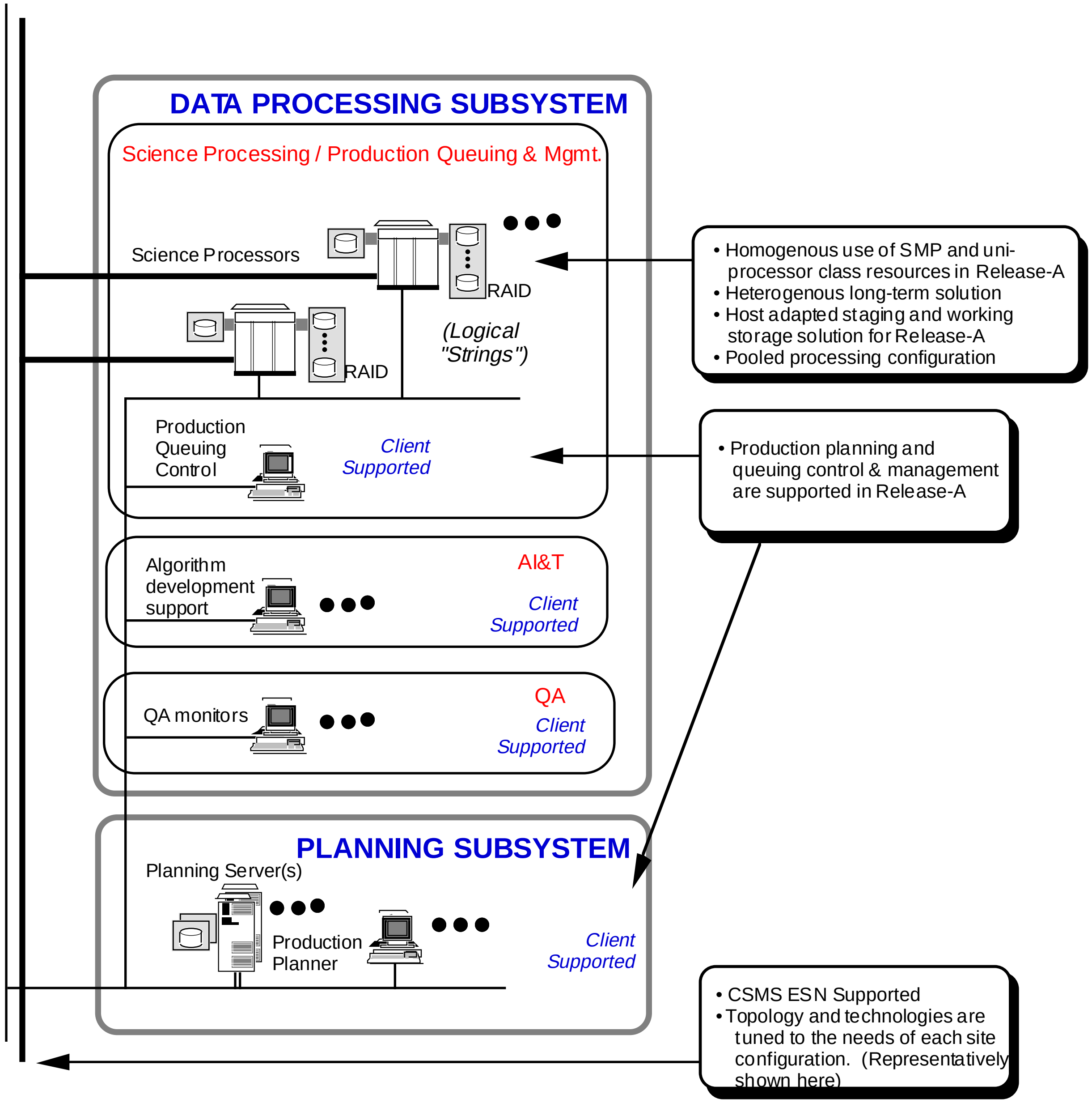

DATA PROCESSING SUBSYSTEM
Science Processing / Production Queuing & Mgmt.
Science Processors
RAID
RAID
(Logical "Strings")
Production Queuing Control
Client Supported
Algorithm development support
AI&T
Client Supported
QA monitors
QA
Client Supported
PLANNING SUBSYSTEM
Planning Server(s)
Production Planner
Client Supported
• Homogenous use of SMP and uni-processor class resources in Release-A
• Heterogenous long-term solution
• Host adapted staging and working storage solution for Release-A
• Pooled processing configuration
• Production planning and queuing control & management are supported in Release-A
• CSMS ESN Supported
• Topology and technologies are tuned to the needs of each site configuration. (Representatively shown here)

Figure 8-1. Topology of Data Processing and Planning Subsystems

This section provides the HWCI design rationale for the proposed high end SMPs (SMP-H) for LaRC, EDC, and GSFC and the uniprocessor workstations for MSFC The processing complement for MSFC is designed and sized for the TRMM mission. The SMP supports sequential processing, in addition to symmetric and distributed memory parallel processing paradigms. This platform is analogous to "multiple workstations in a box," providing a coherent shared memory and containing identical processors with a single operating system. The cost/performance tradeoff (see Section 8.1.1.2.1), shows SMPs provide the best price/performance with more growth capacity. SMPs also have very good scalability, relatively high I/O bandwidth (e.g., 1.3 GB/s Peak), and very high interconnect bandwidths. A market survey has indicated that future SMP performance is increasing at a faster rate than conventional MPPs.

8.1.1 Key Trade-off studies and Prototypes

8.1.1.1 HWCI Alternatives

Various candidate processor classes were evaluated for IR-1 and Release A including uniprocessor workstations/servers, workstation farms/clusters, vector supercomputers, and parallel processors. The parallel processor class includes: (1) Low-end symmetric multi-processors (SMP), or SMP-L, (up to 8 CPUs), (2) High-end SMPs, or SMP-H (up to 12 or more CPUs), (3) Massively Parallel Processors (MPP) (more than 64 processors), and (4) SMP clusters, consisting of more than one fully populated SMP-H. These platform classes are briefly described below:

- Uniprocessor Workstation/Servers—This platform class is a low cost commodity item configured with a single CPU often controlled directly by the user. By itself, the workstation has limited scalability.
- Workstation Farms—A workstation farm is a cluster of workstations connected via Local Area Network (LAN). This connectivity provides excellent scalability. However, low network bandwidth and high latency are limitations of this class.
- Vector Supercomputers—Vector supercomputers were considered, but for IR-1 and Release A their cost and suitability did not match the processing requirements for these early releases. This category is a potential platform for Release B and will be re-evaluated at that time.
- Symmetric Multiprocessors (SMPs)—The SMP class provides excellent cost/performance, graceful degradation, and good scalability. This class is selected on the basis of a cost effective solution that meets both performance and scalability requirements. Interconnect bandwidths within the SMP are high. The SMP can process sequential science algorithms with the capability to transition to parallel processing of these algorithms.
- Massively Parallel Processors (MPPs)—Massively parallel processors (MPPs) contain numerous processors (usually more than 64) MPPs provide very high speed interconnects and can provide multiple I/O channels.

The platform classes which suit the requirements for Releases IR-1, A and the initial phases of B are the SMP for LaRC, EDC, and GSFC and the Uniprocessor Workstation/Server for MSFC (See Table 8-1). Key trade studies were performed (See Section 8.1.1.2) to support the selection of hardware platform classes. The SMP is proposed because it offers good price/performance, handles parallelization of science software, is scalable and evolvable. Processing, disk storage, and input/output requirements for each DAAC were based upon January 1995 technical baseline data and evaluated with static analysis and dynamic modeling.

DAAC unique characteristics of the processor platforms are provided in the DAAC Unique Volumes (for GSFC: DID-CD-305-014, for LaRC: DID-CD-305-015, for MSFC: DID-CD-305-016, and for EDC: DID-CD-305-017) for the operational sites (i.e., MSFC, and LaRC) and development sites (i.e., GSFC and EDC) for Release A. Recommended quantities, configurations as well as vendor/model identification (candidate) are discussed for each site in detail.

Table 8-1. Platform Recommendation for IR-1 and Release A

Site	Platform Class	Supporting
LaRC	SMP-H (Minimum) + SMP-H(s) + Uniprocessor workstation (Queuing)	IR-1 Rel A Rel A
MSFC	Uniprocessor workstation +Uniprocessor workstation +Uniprocessor workstation (Queuing)	IR-1 Rel A Rel A
EDC	SMP-H (Minimum) No Change in Science Processing +Uniprocessor workstation (Queuing)	IR-1 Rel A Rel A
GSFC	SMP-H +CPUs +Uniprocessor workstation (Queuing)	IR-1/Rel A Rel A Rel A

8.1.1.2 Key Trades

The SPRHW design analysis for PDR was accomplished through a number of efforts: cost/performance trade-off analysis, key design trade studies, prototyping, and joint analysis of requirements with the AHWGP.

8.1.1.2.1 Cost/Performance Tradeoff

A cost/performance trade-off has been performed for the various candidate processor platform classes for the ECS Release-A sites discussed in Table 8-2. Representative vendor average list prices are normalized to peak MFLOPs performance for each of the platform classes. The SMPs are broken down into two categories: SMP-L (1 to 8 CPUs) and SMP-H (2 to 64 CPUs). The main distinction between the high- and low-end SMPs are processing power, scalability, and cost. SMPs are further classified as minimum and maximum configurations, where minimum refers to the least number of CPUs configured for the SMP (e.g., 1 CPU for SMP-L, and 2 CPUs for SMP-H) and maximum refers to a fully populated SMP. The Dollars-Per-MFLOPs is taken as an average normalized cost of the minimum and maximum SMP configurations.

Considering the AHWGP required processing performance capacities as a constraint, the most cost-effective processor platform selection for the LaRC, and GSFC DAAC sites is the high-end SMP (SMP-H). Although, the low-end SMP cost/performance ratio is less than the high-end, this class of SMP cannot satisfy the MFLOPs nor the scalability for LaRC and GSFC. The SMP-H is approximately 23% and 60% of the normalized cost (price/performance) of the Vector Supercomputer and the MPP, respectively. Comparison of price/performance is actually more favorable for the Release A maximum configuration SMP-H. In addition to cost/performance factors, suitability, operation in a distributed computing environment, and scalability to Release B are important factors in selecting a particular platform.

The AHWGP required MFLOPs capacities by release and DAAC site are summarized in detail within Section 8.1.2. Static and dynamic analysis overviews are provided. Given the analysis performed to date, processing requirements for EDC can be satisfied by a low end SMP (SMP-L) for IR-1 and Release A. However, it can be shown that to achieve the scalability for Release B, it is more cost effective to procure a SMP-H. The MSFC processing requirements can easily be satisfied by an uniprocessor workstation class. The average list price of the workstations surveyed are less than 1/3 the cost of a minimum configuration SMP-L.

Table 8-2. Processing Support Activities for the Release A ECS Sites

DAAC	Release	Activity
LaRC	IR-1 Rel A	AI&T Science Processing operations
MSFC	IR-1 Rel A	AI&T Science Processing operations
EDC	IR-1 Rel A	AI&T AI&T
GSFC	IR-1 Rel A	AI&T AI&T

This sizing effort was accomplished in conjunction with the performance requirements provided within the January 1995 CDR Technical Baseline (covering IR-1 and Release A). This technical baseline is derived from the set of January 1995 AHWGP data. Performance and capacity requirements are derived from the AHWGP data. A roll-up summary of AHWGP requirements are provided within the SDPS Requirements Specification for the ECS Project (Appendix E, 304-CD-002-001). A comparison of science processor vendor platforms is made in the Production Platform Families for the ECS Project technical paper (440-TP-007-001). Platform selection is based upon price/ performance, suitability, operation in a distributed computing environment, and scalability. Scalability is a crucial factor since performance and capacity requirements increase drastically in Release B.

8.1.1.2.2 Trade Studies

The "Trade-off Studies Analysis Data for the ECS Project" document (Reference: 211-CD-001-001) provides an overview of the related trade studies which are briefly described below.

- Distributed and Parallel Processing
- Production Topologies
- Production Platform Families

A *Distributed and Parallel Processing Trade Analysis* was performed examining the benefits of distributed and parallel computing. The trade studies various processing alternatives for ECS science algorithms and provides up-to-date information on processing technologies. This trade analyzes the applicability of using OSF/ Distributed Computing Environment (DCE), SMP, DMP (including workstation cluster), and MPP for ECS science software.

A *Production Topologies Trade Analysis* examines the advantages and disadvantages of distributing processing tasks from one or more instruments across one or more processing strings. The resulting recommendation will provide a cost effective way of distributing processing to maximize throughput, minimize data movement, and provide and retain the flexibility to evolve with changing processing requirements. This trade analyzes physical (not logical) processing topologies that can impact hardware requirements, overall performance, network capacity, throughput, and staging storage. Recommendations for hardware selection based on cluster optimization alternatives for Release B and beyond are made.

A *Production Platform Families Design Trade* study has been performed to recommend one or more Science Processing HWCI platform processor class(es) based on the Technical Baseline/AHWGP data, scalability, risk, and cost. This trade provides the basis and rationale for Science processing HWCI hardware class recommendation for IR-1 and Release A with projection to Release B. The recommendations resulting from this study are the basis for the Data Processing procurement process scheduled for early 1995.

Three major considerations are examined along with quantitative selection criteria to come up with the final platform recommendations.

- Phased performance requirements
- Algorithm development, test and maintenance costs
- Architecture design impacts

This trade study is being conducted in two phases: (1) Static analysis (spreadsheet analysis) using the November 1994 baseline and (2) Static analysis and dynamic modeling using the January 1995 baseline. Sizing of the processing platforms for Release A has been updated reflecting the new AHWGP data incorporated into the January 1995 Technical Baseline and completion of the static and dynamic modeling. This trade study is being re-issued incorporating the updated set of results.

In addition, further refinements to the baseline through the AHWGP (especially for areas including: QA, reprocessing, and Release B sizing changes) will result in repeated study analysis throughout the Release A CDR phase and beyond. The analysis will be further refined based on dynamic model results. Further detail regarding AHWGP data as applied to this design is discussed in Section 8.1.2.

The AHWGP data provides the following classes of data by epoch and DAAC sites for each PGE:

- Volume at Initiation and Completion (MB)
- Staging and Destaging I/O (MB)
- Total I/O Requirements (MB)
- CPU Requirements (MFLOs)
- Input and Output Files per Execution
- Activations per Day

The following processing and I/O requirements are derived from AHWGP data in a static analysis:

- Processing (MFLOPs)
- Host Attached Backplane I/O
- Network I/O

Relationships of the above latter set of processing and I/O requirements are described in Section 8.1.2.1.

In addition to the MFLOP processing requirements, the I/O bandwidth and disk volume capacities are calculated using the results from static modeling and "time averaged" for epochs e (IR1/Release A time frames) and k (Release B/C timeframes). The peak disk volume was derived from the AH-WGP data by multiplying the volume at initiation by a factor of 2 (2 days worth of staging capacity).

Dynamic modeling was performed for the Release A technical baseline two shift operation (i.e., 16 hours a day, 7 days a week) at LaRC for epoch e to evaluate performance and capacity requirements. Release B was evaluated for 24 hours a day, 7 days a week according to the January 1995 baseline. MSFC is a one shift operation and is similarly modeled.

This information was then analyzed (for each DAAC and instrument) and the following platform classes are recommended: SMP configurations for LaRC, EDC and GSFC, and uniprocessor workstation/server configurations for MSFC. The EDC and GSFC are provided hardware to support AI&T activities for IR-1 and Release A. The SMP solution is viable for IR-1 and Release A (and potentially Release B) because it offers a very flexible platform in which applications can run in either a serial or parallel mode and the SMP, by definition, is a reasonably scalable system. It also provides fast channel communications and is easy to administer.

8.1.1.2.3 Prototype Studies

Prototyping representative science algorithms at the ECS Science and Technology Laboratory (STL) using processing alternatives provided input and rationale for the selection of platform classes for the Science Processing HWCI. The Science Software Execution Prototype used science algorithms (e.g., Pathfinder, AVHRR/Land, SSM/I, SeaWinds) to study applicability of various processing alternatives (DCE, SMP, DMP/Workstation cluster, and MPP). The features of this prototyping effort incorporated distributed computing of Pathfinder AVHRR/Land using OSF/DCE on physically distributed workstation cluster. Multiprocessing, using SMP, DMP/ workstation cluster, of SSM/I and SeaWinds using automatic parallelization tools were demonstrated. SDP Toolkit performance studies were also conducted. The Science Software Execution Prototype also provided inputs for science software portability issues (e.g., 32 bits vs. 64 bit architectures). New processing technologies were revealed including architectures and software tools. The science processing prototyping activities provided hands-on experience with these newer technologies and also the rationale for recommending the most appropriate hardware for the DAACs.

8.1.2 Sizing and Performance Analysis

The purpose of this sizing and performance analysis is to provide the basis for sizing the SPRHW science processors (i.e., MFLOPs and I/O) and its Host attached disk storage, data server disk storage, RAM complements and the network bandwidths between Data Processing and Data Server. This analysis focuses on performance and capacity requirement Release A two shift operation (16 hours a day/7 days a week) at LaRC and compares resources required for 3 shifts (i.e., 24 hours a day). (It should noted that the PDR sizing was based on 24 x7 hours, and the 16 x 7 for LaRC and 8 x 7 at MSFC constitutes a baseline change which affects subsystems sizing). Release B performance and capacity requirements are analyzed to understand the extent of scalability and the migration path from Release A to Release B. Sizing analysis for the Production Queuing component is based on sizing analysis performed within the Planning Subsystem. The sizing of the workstation/servers is based on similar analysis of transaction requirements (as a function of AHWGP PGE invocation needs) and COTS software loading.

Processing requirements discussed in this Release A CDR volume are based upon the January 1995 Technical Baseline/AHWGP data. New AHWGP data has been provided in June 1995 that affects primarily MODIS data (i.e., level 3 MODIS) and to a lesser extent, CERES and MISR data. The June data as of the date on this paper is being analyzed by the modeling and science teams, but has not been applied to the design and sizing analysis described here. DAO data will be provided in mid-August with the assessment of DAO processing and capacity being reported at the Release B IDR.

Initially, spreadsheet or static analyses results are presented for Release A (Epoch e) and Release B (Epoch k) sites. These are processing and data volume demands on a time average basis of two shift operation (16 x 7) at LaRC and one shift operation at MSFC (8 x 7) for 1x processing. These results are translated into average processing (MFLOPs), Input/output (I/O), and network bandwidth requirements. Dynamic modeling results are also provided for the same Release A and Release B site and corresponding epochs. Resource impacts are compared to three shift (24 hours a day, 7 days a week) at LaRC. Sensitivity analyses of number of CPUs and CPU processing (MFLOPs) level are also evaluated.

8.1.2.1 Static Analysis Results

Static analysis of AHWGP data provides an assessment of processing and capacity demands on a time-average basis for 16 X 7 operation at LaRC and 8 X 7 operation at MSFC. A roll-up summary table of static analysis results based on the AHWGP requirements for the Release A operational sites (LaRC and MSFC) during epoch e (1Q98) is provided in Table 8-3. Epoch e (1Q98), represents the stress case for Release A. In order to account for machine efficiency, processing (MFLOPs) is multiplied by a factor of 4 for an assumed 25% machine efficiency. Phasing factors are not applied to these values (i.e., 1 x).

Table 8-3. Static Analysis Summary Results of January 1995 Baseline Data-Release A AHWGP Requirements

Instrument	Site	Processing (MFLOPS	Back plane I/O (MBps)	Network I/O (MBps)	Archive I/O (MBps)
CERES	LaRC	4,693	1.8	0.6	0.2
LIS	MSFC	5	0.1	0.5	0.1

Processing and static estimates of I/O bandwidth are derived from the AHWGP requirements. Basic data flow between science processors, host attached disk, and data handler working storage is shown below in Figure 8-2. A synopsis of the static analysis key performance parameters is described below. Full details are provided within the trade study technical paper.

S: Staging Volume

D: Destaging Volume

I: Processing I/O

The relationships for processing, I/O, and network bandwidth requirements obtained from the static AHWGP data are given by (example below for LaRC operational timelines):

Processing: MFLOPs = MFLOs x A/57,600   where A is activations over 16 hours

I/O at CPU: I/O at CPU = (2S + 2D + I) x A/57,600

Network Bandwidth: NW BW = (S + D) x A/57,600

MFLOPS, I/O and Network Bandwidth requirements are spread over 8 hours in the case of MSFC.

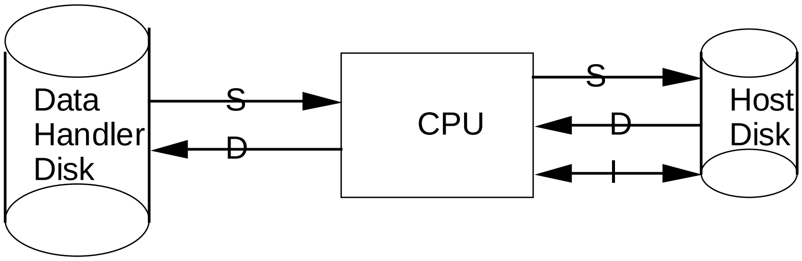


Figure 8-2. Data Flow Block Diagram

Given the TRMM CERES machine rated processing requirement in Table 8-3, and assuming CPUs is rated at 300 MFLOPs, then 16 CPUs, on average, are necessary to meet TRMM CERES processing requirements. The dominant PGE contributing to processing load is CERES 5AF, which is activated once every hour. Backplane I/O, as given in Table 8-3, is relatively low. Static analysis provides results that are time averaged over the operational hours dictated by the baseline, as compared to dynamic modeling results that show peaks.

LIS processing and I/O requirements translate into a workstation class machine. MSFC LIS platforms are sized for the LIS mission in total. All growth factors are applied in the initial delivery to MSFC and are reflected in the sizing given. Numbers on provided capacity are adjusted to include growth for the mission lifetime. LARC, is sized for one calendar year past launch.

Although Release A is the primary focus, Release B performance and capacity provides a "look ahead" to evaluate scalability, and assess the impact of algorithm integration and test (AI&T). A roll-up summary table OF AHWGP Requirements for release B operational sites, for selected instruments, are shown in Table 8-4 for epoch k (3Q99). In order to obtain machine rated MFLOPs, the demand processing is multiplied by a factor of 4 to account for a 25% assumed efficiency.

Table 8-4. Release B Processing, I/O, and Network Bandwidth Static Analysis Requirements for Epoch k (3Q99) (January 1995 Baseline)

Instrument/Site	Processing (GFLOPs)	Backplane I/O (MB/s)	Network I/O (MB/s)
MISR/LaRC	13.84 (46 CPUs)	19.31	6.43
CERES/LaRC	11.32 (38 CPUs)	7.24	2.42
MODIS/GSFC	8.9 (30 CPUs)	121.7	40.72
MODIS/EDC	5.36 (18 CPUs)	194.72	65.14

MISR and CERES instruments at the LaRC site requires the largest processing loads, 13.84 and 11.32 GFLOPs, respectively, for Release B. The purpose of investigating Release B requirement during Release A CDR time period is to project a migration path and satisfy scalability and evolvability to Release B and beyond.

8.1.2.2 Dynamic Model Results

8.1.2.2.1 Dynamic Model Background

The dynamic model (i.e., ECS Performance Model) is a Block Oriented Network Simulation (BONeS) model used in conjunction with the AHWGP data to simulate processing. It is a resource constrained model containing a pool of processors and a pool of disk storage.

BONes is a discrete-event simulation tool for analysis and design of communication networks and distributed processing systems. Components of a distributed processing system are represented by nodes, which have resources associated with them that get allocated as events request them. Standard production of instrument data within the Data Processing Subsystem (DPS) is simulated by the Processing module in conjunction with the Event Driven Scheduler and the Data Handler, where the Data Handler is representative of the Data Server design. The Data Handler stores and retrieves data from the permanent archive, for routing data to the requesting subsystems, and managing tiered storage resources. The scheduler monitors availability of data, requests data to be staged from the data handler to Processing, routes newly created data to the appropriate data handler or processing pool, and initiates execution of a process when all required inputs are present. (It should be noted that the event driven scheduler is not intended to serve as a simulator of the Planning Subsystem.) The Ingest module simulates the Ingest subsystem, which accepts data from external systems and users and contains rolling storage of L0 instrument data.

8.1.2.2.2 Dynamic Modeling Runs

The Dynamic Model (ECS System Performance Model) was run to assess dynamic performance of production processing for both Releases A and B. This model provides data with respect to peak and average resource consumption while simulating a resource constrained environment, which is not possible with static analysis alone. Static or spreadsheet analysis provides 24 hour time averaged data.

Release A—Release A was evaluated for TRMM CERES at LaRC and LIS at MSFC during epoch e. Both two shift operation (16 hours a day, 7 days a week) and three shift operation (24 hours a day) were evaluated for Release A at LaRC.

LaRC Two Shift Operation (16/7) Performance and capacity are evaluated for two shift operation (16 hours a day, 7 days a week) for TRMM CERES at LaRC during epoch e using a maximum of 24 CPUs each rated at 300 MFLOPs. Refer to the DAAC Specific Volumes (references given earlier), for further details on this analysis and how it is applied to derive the specific configurations. The dynamic modeling results are summarized below in Table 8-5.

Table 8-5. Two Shift Operation Performance and Capacity

Max.No.CPUs	MFLOPs/ CPU	Avg. No. CPUs	Host Disk-Max GB	Host Disk-Avg.GB	DH Disk-Max GB	DH Disk-Avg.GB
24	300	10.1	31.5	5.7	19.9	1.1

LaRC Three Shift Operation (24/7) Three shift operation (24 hours a day/ 7 days a week) was also analyzed using dynamic modeling results. Comparison of data processing resources is made of two shift operation (16/7) vs. three shift operation (24/7).

Sensitivity analyses were conducted for 24 hour operation including the following cases:

a. 300, 600, and 900 MFLOPs machines

b. Maximum number of constraining CPUs.

A summary of a subset of the dynamic model processing and capacity results are presented in Table 8-6 Release A TRMM CERES at LaRC.

Table 8-6. Dynamic Model Processing and Capacity Summary for Release A TRMM CERES at LaRC

Max No. CPUs	MFLOPs/ CPU	Avg. No. CPUs	Host Disk-Max GB	Host Disk-Avg GB	DH Disk-Max GB	DH Disk-Avg. GB
18	300	11.54	27.27	5.76	23.25	1.33

Sensitivity to maximum number of CPUs (i.e., 15, 18, 21) for 300 MFLOPs and 24 hour operation was investigated. There was very little sensitivity to average number of CPUs required. However, when 18 CPUs was run for two shift operation, processing could not catch up.

Two shift operation vs. three operation is a trade-off of staffing reduction vs. increasing the number of CPUs. The hardware/cost penalty is 6 additional CPUs for two shift operation. This additional hardware cost is mitigated by the fact that these additional CPUs will be used during release B. Although two shift operation is baseline, this does not preclude processing during unattended operation. Currently, checkpointing is not implemented in the system. If a process stops or a CPU fails, these additional CPUs can provide contingency processing during unattended operation and provide a catchup capability. Nominally, backup capability will be provided by the AI&T science processor in the event of auction science processor failure.

8.1.2.3 Phasing

Application of the phasing factors to the AHWGP processing requirements is a very important step in each DAACs platform class evaluation. Scalability becomes a heavily weighted criteria when the processing at the DAAC ranges from low to high as science software is tested and integrated, instruments are launched and calibrated, and standard production and reprocessing begins. An example of how phasing is applied for CERES processing at LaRC and how the results enter into the platform recommendation is described, and an explanation of phasing factors as applied to standard products follows.

Figure 8-3 summarizes and illustrates how phasing factors are applied to processing capacity based on launch dates. This phasing is applied to each platform and its scheduled launch date and incorporated into the Release schedule.

- **0.3X for L-2 < t < L-1** For pre-launch AI&T starting at launch minus 2 years, AI&T requires 0.3 of the processing estimate at launch during the period 1 to 2 years prior to launch. X is defined as at-launch processing estimate for pre-launch AI&T.
- **1.2X for L-1 < t < L+1** For pre-launch AI&T and system I&T, starting at launch minus 1 year, AI&T and system I&T requires 1.2 times the processing estimate at launch during the year prior to launch. Standard instrument processing requirements begin from launch date and last for the remainder of the life of the instrument. X is defined as at-launch processing estimate for prelaunch AI&T and systems I&T.
- **2.2X for L+1 < t < L+2** For post-launch AIT, standard processing, and reprocessing of data, starting at launch plus 1 year, 2.2 X is required. X is defined as the standard processing estimate for that period.
- **4.2X for t > L+2** For post-launch AI&T, standard processing and reprocessing of data, starting at launch plus 2 years, 4.2 X is required. X is defined as the standard processing estimate for that period.

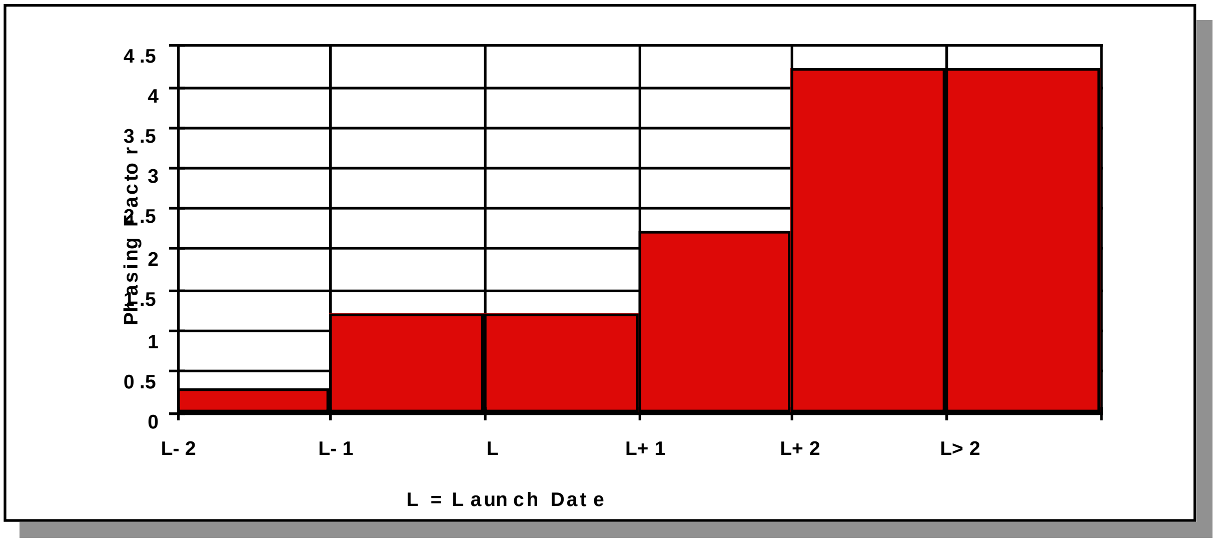


Figure 8-3. NASA ESDIS Phasing Factors

8.1.2.4 RAM Requirements

In general, RAM configuration sizing for SPRHW platforms, from workstations to large SMP class platforms must take into account the following contributing factors:

- COTS / OTS package RAM nominal and peak utilization,
- custom infrastructure and PDPS process utilization estimates (part of the PDPS design),
- and, especially critical to the SPRHW Science Processors, the estimated requirements of the science algorithm PGEs (nominal and peak).

For platforms exclusive of processing, reprocessing and actual AI&T support, it is predicted that the first two items above will provide for the core RAM sizing estimate. For the Science Processors, the third factor consisting of Science PGE requirements is likely to dwarf the ECS COTS/ OTS and custom process requirements.

Tall pole Science PGEs, in terms of processing and/or disk, may or may not translate into tall poles as far at RAM requirements are concerned. The implementation of the PGE is the key. The PGE RAM requirements in the near term for Release-A, will likely come from a combination of one or more of the following types of data:

- Science heritage code estimates and/or actuals (analogous applications to planned PGEs),
- Estimates based on current PGE development,
- Estimates and/or measurements from early AI&T (prototyping and development).

The Science Processor configurations will take these factors into account over time, with better estimates and measurements used over time to refine the configuration. The planned or recommended configurations (with their inherent limitations and associated cost) will have ramifications to the Science Team's engineering as well.

Where RAM information estimates are available, the DAAC Specific Volumes contain descriptive rationale and configuration information.

8.1.3 Scalability, Evolvability and Migration to Release B

Release B was evaluated for LaRC, GSFC, EDC, MSFC, ASF JPL, ORNL, and NSFOC during epoch k. The major processing loads occur at LaRC and GSFC. Both sites are three shift operation (24 hours a day). Table 8-7 provides a summary of the core Release B requirements for hardware support derived from this analysis.

Table 8-7. Dynamic Model Processing and Capacity Summary for Release B at LaRC and GSFC

Site	Max No. CPUs	MFLOPs/ CPU	Avg. No. CPUs	Host Disk Max GB	Host Disk Avg GB	DH Disk Max GB	DH Disk Avg Disk
LaRC/ MISR	48	300	46.4	560.3	534.9	103.3	13.8
LaRC/ CERES AM-1	54	300	36.0	100.1	46.0	103.3	13.8
GSFC/ MODIS	53	300	34.5	83.3	35.7	97.9	15.7

These results in the above table show a dramatic increase in processing and storage requirements for release B. In particular MISR host storage requirements are extremely high (i.e., 560 GB). Additionally, MODIS I/O requirements are very high as can be seen in static analysis. Release B performance and capacity requirements are very much higher than Release A.

High end SMPs are the appropriate choice for LaRC in Release A since the Release B requirements are high and there is need for built in scalability (investment). The purpose of investigating Release B requirements during Release A CDR time period is to project a migration path and satisfy scalability and evolvability to Release B and beyond.

Scalability is the ability to increase processing capacity with minimum impact on both hardware and software. From a hardware standpoint, it is the ability to add on processors; from a software standpoint, it is the ability to provide incrementally better performance with minimum tuning. This ability to expand processing with minimum effect on existing operations can be achieved in several ways, including but not limited to:

- adding new processing clusters
- adding CPUs to existing host computers
- adding computer systems to existing clusters
- adding subnetworks to support additional inter and/or intra-cluster I/O and communications
- technology refresh

The ability to select a processor class which satisfies increasing performance requirements in incremental releases and a spectrum of scientific missions including TRMM, AM-1, and PM-1 is one of the major challenges of the design. Initially, processor performance requirements must satisfy the TRMM mission with CERES and LIS instruments in Release A.

An SMP configuration provides reasonable scalability (up to 32 processors) and is more than adequate for most sites included in Release A. For Release B, the scalability can be extended (primarily at LaRC and GSFC) by employing clusters of SMPs within a high-speed network.

8.1.4 Algorithm Parallelization

The major consideration in selecting the processor class is the ability to run parallel programs especially when large processing requirements are called for Release B. An SMP provides the ability to parallelize programs in shared memory (the easiest paradigm for parallel programs) in shared memory and distributed memory modes. A variety of tools are available to parallelize science software. The ECS Science and Technology Lab (STL) prototyping (Reference: 440-TP-008-001, 194-00569TPW, 194-430-TPW-001) of science processing has demonstrated parallel program development based primarily on parallelization tools.

8.2 HWCI Structure

The SPRHW HWCI consists of two major components that support processing as well as production queuing, monitoring and control. A Release A specific block diagram of the Science Processing hardware is shown in Figure 8-4, which illustrates the two major logical components within SPRHW, and the physical building blocks those components are constructed with.

The block diagram gives a psuedo-physical view of the SPRHW layout and how the physical equipment classes relate and connect to one another. For Release A, the candidate architecture being recommended consists of pooled processing resources interconnected with CSMS provided ESN LANs (subnetworked as necessary to meet site specific throughput requirements). The Production Topologies Trade Study provides a discussion of this analysis and provides a view of the options available for Release B and beyond, which requires a more sophisticated topology at the larger processing sites.

Hardware requirements for the processing strings may range from small or medium single processor workstations, to SMP compute servers, or a cluster of SMPs. Unique algorithm I/O requirements may point to the selection of compute resources that offer the best solutions in terms of I/O subsystem growth and augmentation capability. Depending upon the volume and I/O requirements, staging resources may be clustered, network attached or host attached (dedicated). The two subsections that follow provide details regarding the interconnection between component equipment classes as well as identifying the types of equipment classes required.

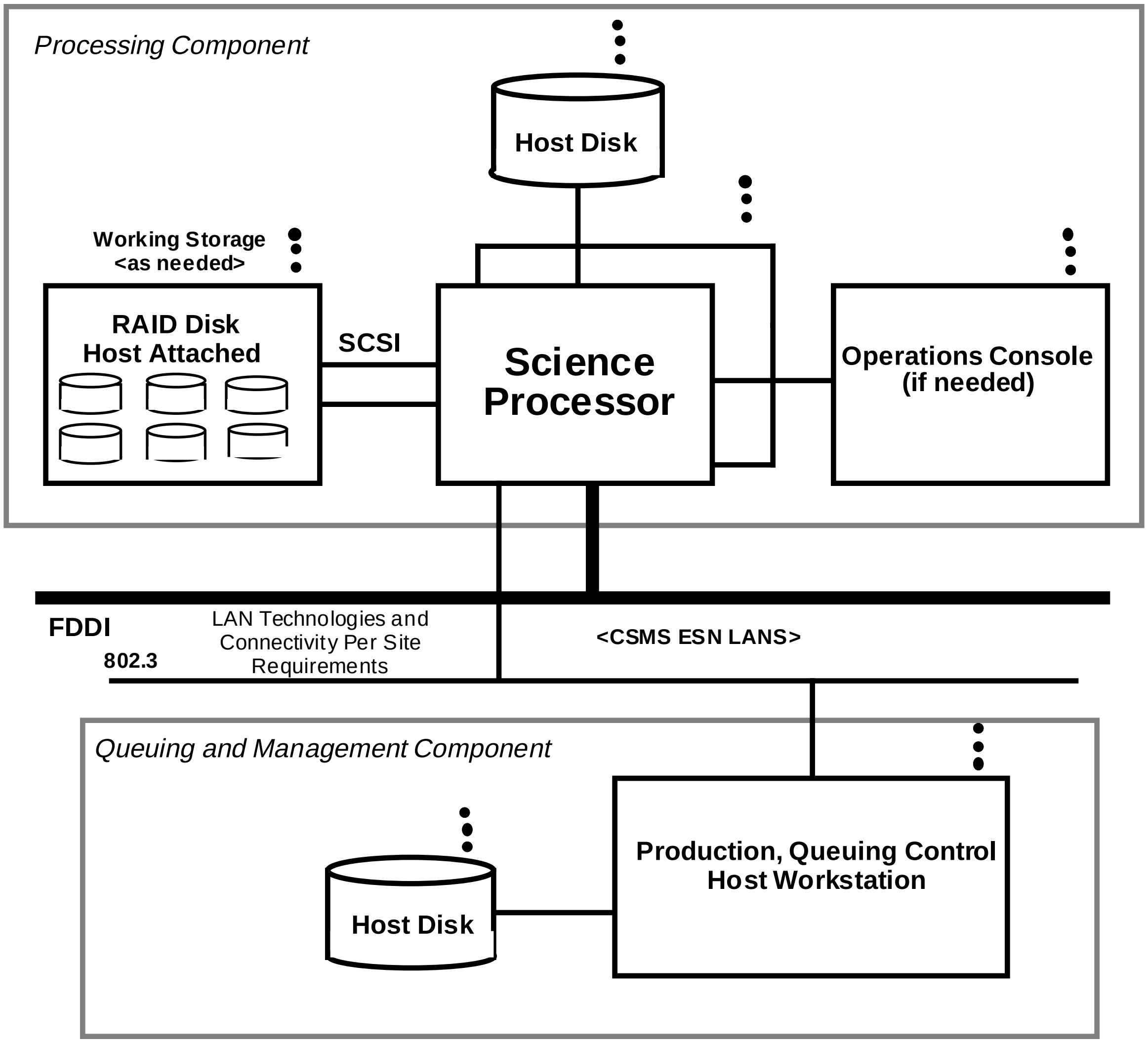


Figure 8-4. Science Processing Block Diagram

8.2.1 Connectivity

The intra DAAC data interfaces will be implemented as follows: disk interfaces will be of the channel type (e.g., SCSI II); control interfaces will be of the network type (e.g., FDDI, ethernet).

The Planning and Data Processing subsystem network connectivity is illustrated in Figure 8-5. Both subsystems will connect directly to the same FDDI ring. Hosts (servers and workstations) will contain single-attached station (SAS) cards and will be connected to an FDDI concentrator, which will in turn be connected to the FDDI switch via a physically wired FDDI ring. (Refer to Section

TBD of Volume 0 for a general description of DAAC networks, and to Section TBD of Volumes X-Z for DAAC-specific topologies.)

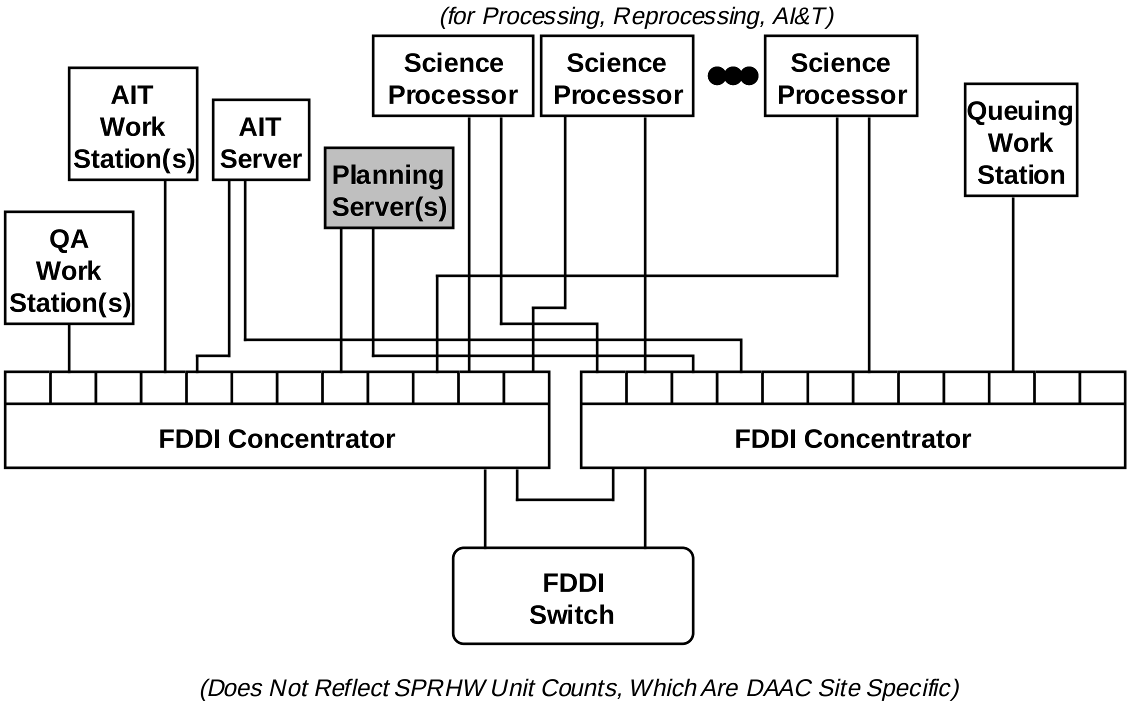


Figure 8-5. PDPS Network Connectivity In Release A (Generic Hardware Units)

8.2.2 HWCI Components

Table 8-8 provides a overview of the two major logical components within SPRHW, and the physical classes of hardware that support them. SPRHW consists of two logical components as shown in the block diagram, Figure 8-4:

- a *Processing Component* that provides the site specific required AI&T, standard, reprocessing and (future) on-demand processing capacity, and
- a *Queuing and Management Component* that provides workstation support for production queuing, monitoring and control (automated and operations support based).

This table provides a generic and site independent view of the components and their physical building blocks. See the DAAC specific volumes for MSFC, GSFC, LaRC, and EDC configurations. Not all sites are in the same state of operational readiness, nor are their core requirements similar, therefore their complement will be different from site to site.

Table 8-8. SPRHW Logical Components and Equipment Classes

Component Name	Class/Type	Comments
Processing	SMP Science Processor (SMP-H, SMP-L) (not at all sites)	• Provides science processing capacity for science software integration and test, standard production, on-demand production (future releases), as well as reprocessing support. • Large and/or small SMP processors with host attached system disk and local console support. • Supports two or more processors. • Configuration and/or inclusion in site topology is based on site processing requirements. • Multiple units provided as needed to support capacity needs and operational needs.
	RAID Disk (Host Attached) (not at all sites)	• RAID storage for staging and destaging of science products. • Support for interim and temporary files in addition to staging capacity for production data sets and dependency datasets. • Host attached in release A (e.g., host attached to SMP-H, SMP-L, etc.). Possible network attached applications for release B and beyond (including shared pooling between DS, Ingest and Data Processing subsystems.
	Host Disk (system disk)	• System disk for O/S and other core functions. • Basic disk in core system configurations.
	Science Uni-Processor Workstation (not at all sites)	• “Small” single processor workstations, providing science processing capacity for science software integration and test, standard production, on-demand production (future releases), as well as reprocessing support. • Configuration and/or inclusion in site topology is based on site processing requirements. • Multiple units provided as needed to support capacity needs and operational needs.
	DBMS Server (Future & instrument processing specific)	• This class of equipment is under investigation for possible support of embedded DBMS applications (future analysis) • Predicted as not applicable for Release-A, but probable application exist for inclusion in H/W topology for Release-B
Queuing and Management	OPS Workstation (small)	• One or more operations workstations, on a site by site basis that support production queuing, monitoring and control. • Provides workstation processing, I/O and disk resources for control and monitoring of the Processing Component production resources discussed above.

8.3 Failover and Recovery strategy

8.3.1 Network Failure Recovery

In general, network failure recovery does not inherently affect the core design of the SPRHW configuration:

- The release A DPS configuration is built primarily on FDDI networks, therefore there is a significant degree of fault tolerance in the physical communications system. Most media failures within the FDDI fabric will not result in any loss of service and no reconfiguration would be necessary in these cases (due to the basic nature of FDDI).

- Given the inherent fault tolerance of FDDI, it was not required to have multiple physical communications paths to each host. Hosts within the DPS will be generally be SAS adapted.
- Failures within the hub/switch fabric will be handled by the CSMS subsystems and do not require special consideration within the physical architecture of the DPS. The core design discussed within this section take RMA analysis and requirements into account.

Within the CSMS ISS configuration for PDPS, there are three types of network failures that may affect the PDPS subsystem. If the FDDI cable between a host and the FDDI concentrator is severed or damaged, then a new cable would need to be installed. No other configuration would be required. If an individual port on the FDDI concentrator fails, then the attached host must be moved to another port, again with no other configuration required. Finally, if the entire concentrator fails, then it will have to be replaced, which can be done rapidly since the units require very little configuration.

Note that the above failures result in service interruption only to the workstations. Since all servers/ processors are attached to two hubs, they will communicate as normal in the event of a cable or concentrator fault, and the applications will be unaware of and unaffected by the event.

8.3.2 Data Processing Subsystem Failure Recovery

The Data Processing Subsystem function has the following RMA requirements for the Product Generation function:

(1) Availability: 0.96

(2) Mean Down Time: <4 Hrs.

This function is supported by the Science Processor(s) and RAID disk storage. These RMA requirements are met by the Product Generation function of the data processing subsystem. An analysis is provided in Availability Model/Predictions for the ECS Project, 515-CD-001-003.

The failover and recovery strategy for the Data Processing System is based upon software control by the Planning Subsystem, and Queuing, and in conjunction with MSS. MSS orchestrates operations management control and Planning and Queuing schedules jobs, and keeps track of available resources.

The AI&T processors are a part of SPRHW processing pool. AITHW just provides workstations for management and control of the test process. The strategy employed is to make best use of available resources. for processing, reprocessing, test, etc., as needed. It is a resource pool, configured in clusters. Failover is handled by the production queuing software as well as the planning software and is controlled in conjunction with MSS. No special hardware is provided to facilitate this. SPRHW is sized and configured to support processing, reprocessing and AI&T. There is no special hardware provision to expedite switchover, it is software and operations procedure based.

Backup of the production science processor(s) will be provided by the SPRHW science processor supporting Algorithm Integration and Test. (AI&T) at LaRC for Release A. The science processor supporting AI&T is a development platform that has a lower priority than the production science processors. In the event of a power supply failure, this component will be replaced by a spare, where the mean down time of less than 4 hours and a production computer availability of 0.95 can be met. This is also true for the MSFC configuration. Host Disks are critical, therefore the configurations include RAID disks for LaRC and MSFC operational sites and GSFC and EDC development sites.

If a CPU fails, the science processor can continue to operate using the other available CPUs in the SMP with minor degradation in utilization. The failed CPU can be spared at the appropriate maintenance event, which is typically scheduled like any other activity by the Planning Subsystem.

Disk Mirroring and Primary and Secondary Servers

These recovery and failover strategies apply primarily to the Process Queuing component within the SPRHW design. The component makeup of the Science Processing component does not utilize these strategies. Similar strategies are applied to the Planning Server within the Planning Subsystem discussed in Section 5.0. For a full description of these strategies, and how they support failover and recovery of Planning (and Queuing) database servers, see Section 5.3.

8.4 Data Processing Hardware Provided Capacity

Section 8.1.2 provided summary information with respect to the AHWGP requirements, static and dynamic analysis as well as phasing of the resultant capacities. Provided capacity, supplied in the sizing discussed in that section, results in Release-A DAAC specific configurations supporting operations at LaRC and MSFC, and early AI&T at GSFC and EDC. The DAAC Specific Volumes for these sites provide a complete description of the provided capacities, associated rationale and the resultant recommended configurations. The capacity sizing information, discussed in Section 8.1.2 as part of the overview on key trades is not repeated here. The following items are noteworthy:

- ESDIS phasing factors and machine efficiencies are applied,
- LaRC required MFLOPs capacities at IR-1 was projected ahead to 6 months after launch
- GSFC MFLOPs at IR-1 assumes 50% of 0.3 X phasing factor
- LaRC disk volume estimate based on DAAC manager estimate. EDC disk volume estimate based on instrument team input.
- LaRC provided capacities supports 2 shift operation for Release A.
- MSFC provided capacities supports 1 shift operation for Release A.

Selected platform classes and corresponding vendors for the operational sites for Release A are:

- LaRC: High End SMP
- MSFC: Workstation Uniprocessor Class

For LaRC the SMP-H recommendation meets immediate performance requirements and was selected because it is a minimum risk approach, providing a smooth transition from IR-1 to Release A, requiring minimal regression testing. This solution provides the reasonably good scalability required for the releases considered.

The recommendation for the MSFC, is a uniprocessor workstation or server. A representative uniprocessor supporting 125 MFLOPs can readily support each site for the life of the contract based on the processing requirements. Similarly, a second uniprocessor was provided in Release A to support fail-soft capability.

Selected platform classes and corresponding vendors for the development sites performing early I/F and AI&T testing: are:

- GSFC: High End SMP
- EDC: High End SMP

The recommended platform for GSFC is a high-end SMP (SMP-H) configuration to support MODIS for IR-1 and additional CPUs provided for the Release A timeframe. GSFC supports AI&T in Release A and, therefore, does not require a backup processor.

The recommended platform for EDC is a SMP scalable to meet phased performance for AIT/I&T, standard processing and reprocessing. The processing at EDC is split between the ASTER and MODIS instruments. Considering the scalability, the EDC processor platform is sized from a minimum SMP-H for IR-1 to a large configuration to support the end of contract requirement. Since EDC only supports AI&T for Release A, a second platform is not provided.

8.5 Pertinent References

1. The ECS Science and Technology Lab (STL) Prototyping, 440-TP-008-001, 194-00569TPW, 194-430TPW-001

9. QAHW - Algorithm Quality Assurance HWCI

9.1 HWCI Overview

Algorithm Quality Assurance HWCI (AQAHW) is the second of three HWCIs of the Data Processing Subsystem. This HWCI contains hardware resources to support DAAC operations and users performing planned routine QA of product data. While the actual processing resources are included within the SRPHW, this HWCI provides the basic facilities to control and enable QA as a process within each DAAC facility (as needed per site specific science and operational policies).

This HWCI does not necessarily support the other primary forms of QA currently envisioned as supported by ECS: in line QA and SCF based QA. These forms of QA are supported by other HWCIs as well as subsystems within the SDPS. (Capacity to directly support inline QA is supported by the sizing of the SPRHW HWCI discussed in Section 8.0.)

9.2 HWCI Design Rationale

QA processing requirements are currently being jointly evaluated with the investigative teams. Current operational assumptions include DAAC QA process performed at the sites in conjunction with SCF-based QA. The current design baseline thus includes local QA monitors.

These local QA monitors are actually similar to Science User workstations equipped with core Client subsystem functionality. These QA monitors, with one or more at a DAAC site, act as QA Clients to the Data Processing components and the Data Server. The current OPS concept, under investigation, assumes that these QA monitors host Algorithm/Science Team supplied processes, which use a subset of the ECS servers provided primarily by the Data Server and Client subsystems to "pull" production data sets under the subscription mechanism. Depending upon the operational requirements of the DAAC and the Science Teams (under investigation), this QA can involve full or statistically determined "pull" loads from the production system through the Data Server. Drastically different local data flows can result based on these requirements resulting in differing QA monitoring workstation and communications configurations. These decisions are DAAC specific.

9.3 HWCI Structure

A block diagram of the Algorithm QA hardware is shown in Figure 9-1. At a minimum, the hardware is configured for general user and subscription use (client support).

This HWCI contains the hardware necessary to support *DAAC operations users* performing planned routine QA of product data (this occurs in parallel, and in conjunction with, routine automated QA supported by the algorithm and/or QA at the SCF performed by the science teams). This HWCI consists of QA monitors and workstations ranging from X-Terminals, to small user workstations, to medium or large graphics workstations. The complement is site dependent and is a function of the classes of production performed.

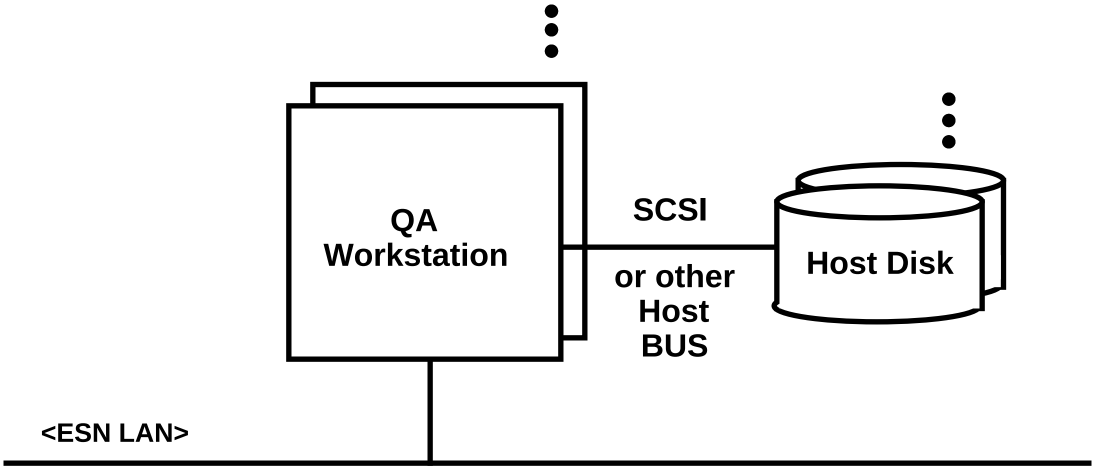


Figure 9-1. Algorithm QA Block Diagram

The need for visualization support will be explored as product specific QA processes and requirements are worked jointly with the instrument teams. In later releases, the QA HWCI accesses the services of the Data Server to perform subsetting. In general, QA positions within the component are clients which may rely on Data Server subscription services to perform their routine missions. This HWCI is supported by the Interoperability Subsystem’s and Client Subsystem’s services. Low to medium bandwidth (e.g., 802.6, FDDI) LANs provided by CSMS provide the connectivity with the Data Server, Ingest and Planning subsystem components as well as the Data Processing subsystem’s string resources.

DAAC unique characteristics of the processor platforms are provided in the DAAC Unique Volumes (for LaRC: DID-CD-305-015, for MSFC: DID-CD-305-016) for the operational sites (i.e., MSFC, and LaRC) for Release A. As warranted by the site OPS concepts and requirements, recommended quantities and configurations are discussed for each site in detail.

10. AITHW - Algorithm Integration & Test HWCI

10.1 HWCI Overview

The Algorithm Integration & Test HWCI (AITHW) provides the hardware resources to support DAAC operations users performing (and/or assisting instrument team members with) science Algorithm Integration and Test (AI&T) (science software), systems validation and integration and test. It is important to note that this HWCI provides the workstation based operations support hardware and necessary server hardware, while the prime science software integration and test *capacity* is provided within the SPRHW HWCI (the science uni-processor and/or SMP hardware). No science processors are supplied by this HWCI to the DAAC configuration. Thus, the AITHW HWCI just provides the operations support workstations to allow DAAC personnel to configure, control and manage the AI&T processes engaged on the target science processors.

10.2 HWCI Design Rationale

10.2.1 Key Trades and Analyses

There are no key trades and analysis especially performed for the sizing and specification of this hardware other than the site specific operations analysis currently underway for all segments and all sites. Workstation and server types and counts will be matched to the needs of the DAAC site and the system release in question.

10.2.2 Scalability Strategies

Since the contents of this HWCI consists only of operations workstations and small servers, there is little need for elaborate scalability strategies:

- As AI&T operations expand due to more processing and/or additional mission support, additional workstations can be added to the network topology.
- To support short term increases in AI&T activity, other operations workstations, already within the configuration can be used since the ESN LANs provide full inter-connectivity. Workstations from the Planning Subsystem and/or the AQAHW HWCI could be used temporarily. (This is true for operations workstations in general, since most of them are not "dedicated" for one and only one purpose, but may support windows into many operational dialogs simultaneously.)

10.3 HWCI Structure

A block diagram of the AITHW HWCI is depicted in Figure 10.3-1. The major elements include operator workstations and small servers equipped with a nominal quantity of host attached disk. Significant disk allocations are not required within this HWCI since the processes that are controlled through the use of these workstations will execute on science processors, provided by the SPRHW HWCI, and configured with sizable resources. However, minimal local AI&T tasks may be accomplished locally on the workstation.

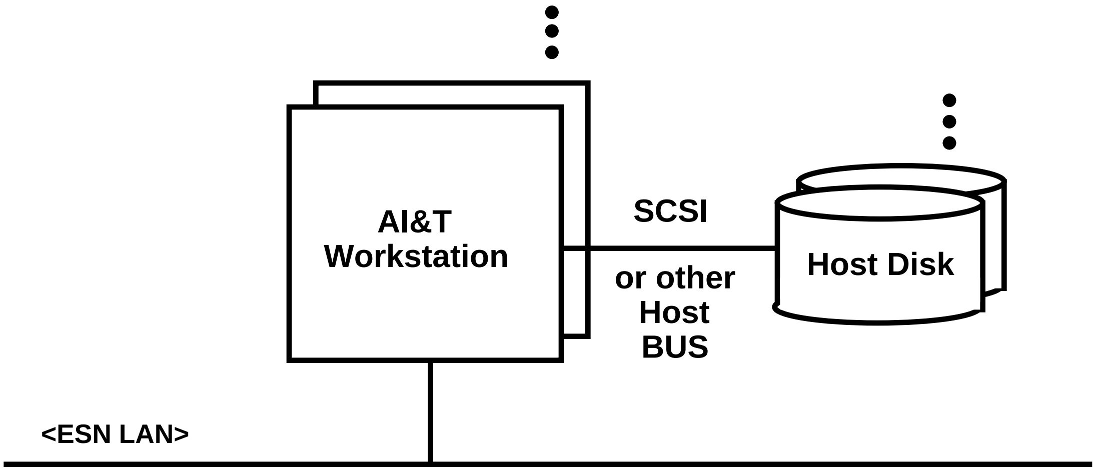


Figure 10.3-1. Algorithm Integration & Test Block Diagram

While equipped with workstations, this HWCI is supported by the Interoperability Subsystem's and Client Subsystem's services. Low to medium bandwidth (e.g., 802.6, FDDI) LANs provided by CSMS provide the connectivity with the Data Server, Ingest and Planning subsystem components as well as the Processing subsystem's string resources. The number and type of workstations is site specific.

10.3.1 Connectivity (Classes of Interfaces)

The intra DAAC data interfaces will be implemented via ESN LANs and host adapters.

10.3.2 HWCI Components

This HWCI does not include any logical components due to its simple implementation. The HWCI will include one or some number of the following physical equipment classes:

- Operations workstations with a nominal allocation of host attached disk,
- AI&T server with host attached disk

Appendix A. Requirements Trace

The Interim Release 1 (Ir1) and TRMM Development (Release A) Level 4 requirements listed in the following table reflect the state of the RTM database on July 15, 1995.

Table A-1. Requirements Trace (1 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-20040	The PRONG CI design and implementation shall have the flexibility to accomodate Processing expansion up to a factor of 3 in its capacity with no changes to the design, and up to a factor of 10 without major changes to its design. Such expansion in capacity or capability shall be transparent to existing algorithms or product specifications.	PRONG CI; SPRHW CI;
S-DPS-20100	The PRONG CI shall request information about the health and availability of a Hardware Resource by using a Systems Management Subsystem (MSS) provided Resource Management API (Application Program Interface).	DpPrResourceManager; MsManager; COTS;
S-DPS-20120	The PRONG CI shall inform the MSS using a MSS provided Fault Management API when a fault attributed to a MSS managed resource has occurred.	PRONG CI; MsManager; COTS;
S-DPS-20130	The PRONG CI shall provide Fault Management data to the MSS using a MSS provided Fault Management API.	PRONG CI; MsManager; COTS;
S-DPS-20140	The PRONG CI shall provide Performance Management data to the MSS using a MSS provided Performance Management API.	PRONG CI; MsManager; COTS;
S-DPS-20160	The PRONG CI shall provide Accountability Management data to the MSS using a MSS provided Accountability Management API.	PRONG CI; MsManager; COTS;
S-DPS-20170	The operations staff shall have the capability to modify the configuration of Data Processing subsystem Hardware resources.	DpPrCotsManager; COTS; DpPrPge; DpPrExecutionManager; DpPrResourceManager; DpPrDataManager; MsMgCallBacks; MsManager;
S-DPS-20180	The PRONG CI shall provide an interface to support the modification of the configuration of the Data Processing subsystem Hardware resources.	DpPrCotsManager; COTS; DpPrPge; DpPrExecutionManager; DpPrResourceManager; DpPrDataManager; MsMgCallBacks; MsManager;

Table A-1. Requirements Trace (2 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-20190	The PRONG CI shall have the capability to modify the configuration of the Data Processing subsystem Hardware resources.	DpPrCotsManager; COTS; DpPrPge; DpPrExecutionManager; DpPrResourceManager; DpPrDataManager; MsMgCallBacks; MsManager;
S-DPS-20210	The PRONG CI shall have the capability to determine the Operational state of a Hardware or Software component.	PRONG CI; MSS ; COTS;
S-DPS-20220	The operations staff shall have the capability to request a Data Processing Subsystem Resource Utilization Report from the MSS based on time span, resource classification, or operational role.	DpPrResourceManager; MSS; COTS;
S-DPS-20230	The PRONG CI shall provide Security Management data to the MSS using a MSS provided Security Management API.	PRONG CI; MSS; COTS;
S-DPS-20240	The PRONG CI shall provide Scheduling Management data to the MSS using a MSS provided Scheduling Management API.	PRONG CI; MSS; COTS;
S-DPS-20330	The PRONG CI shall accept a Cancel Data Processing Request message to delete a Data Processing Request from the Processing Queue.	DpPrExecutionManager, PlDpr; DpPrScheduler; DpPrDataManager; DpPrCotsManager; Cots;
S-DPS-20340	The PRONG CI shall reject a Cancel Data Processing Request if the Cancel Data Processing Request is received from an unauthorized source.	DpPrExecutionManager, PlDpr; DpPrScheduler; DpPrDataManager; DpPrCotsManager; Cots;
S-DPS-20400	The PRONG CI shall accept a Data Processing Request (DPR) that requests the execution of a PGE.	PlDpr; PlPge; DpPrDataManager; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-20410	The PRONG CI shall validate the information associated with the Data Processing Request.	PlDpr; PlPge; DpPrDataManager; DpPrScheduler;
S-DPS-20420	The PRONG CI shall reject a Data Processing Request if the Data Processing Request is received from an unauthorized source.	PlDpr; PlPge; DpPrDataManager; DpPrScheduler;
S-DPS-20430	The PRONG CI shall take a pre-determined error recovery action if the PGE identified in the Data Processing Request is not available for execution.	PlDpr; PlPge; DpPrDataManager; DpPrScheduler;
S-DPS-20440	The PRONG CI shall take a pre-determined error recovery action if the level of validation required for execution in the Data Processing Operational Environment has not been attained by the PGE version identified in the Data Processing Request .	PlDpr; PlPge; DpPrDataManager; DpPrScheduler;

Table A-1. Requirements Trace (3 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-20460	The PRONG CI shall take a pre-determined error recovery action if the resource which maintains the input data is not available for data staging.	DpPrDataManager; DsClESDTReferenceCollector; DsClRequest; DsClCommand; PlDPR; PlDataGranule; GlCallBack;
S-DPS-20470	The PRONG CI shall take a pre-determined error recovery action if the resource identified as the recipient of the Output Data is not available for data destaging.	GlCallBack; DpPrDataManager; COTS;
S-DPS-20480	The PRONG CI shall take a pre-determined error recovery action if the computer resource required to execute the PGE is not available.	DpPrResouceManager; GlCallBack; COTS;
S-DPS-20490	The PRONG CI shall queue only validated Data Processing Requests	PlDpr; PlPge; DpPrDataManager; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-20500	The Processing shall queue the Data Processing Request using the Priority Information associated with the Data Processing Request.	PlDpr; PlPge; DpPrDataManager; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-20510	The PRONG CI shall respond to the source of the Data Processing Request with a Data Processing Request Response upon the completion of validation and queue processing.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-20520	The Data Processing Request Response shall include a reason for rejection if the Data Processing Request was rejected.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-20600	The PRONG CI shall be able to determine what data required for PGE execution needs to be staged.	PlDpr; DpPrScheduler; DpPrDataManager; DpPrDataMap; PlDPR; PlDataGranule; GlUR;
S-DPS-20610	The PRONG CI shall be able to determine that an ECS Data Product required for PGE execution requires staging.	PlDpr; DpPrScheduler; DpPrDataManager; DpPrDataMap; PlDPR; PlDataGranule; GlUR;
S-DPS-20620	The PRONG CI shall be able to determine that the metadata associated with a ECS Data Product required for PGE execution requires staging.	PlDpr; DpPrScheduler; DpPrDataManager; DpPrDataMap; PlDPR; PlDataGranule; GlUR;
S-DPS-20630	The PRONG CI shall be able to determine that an Ancillary Data Product required for PGE execution requires staging.	PlDpr; DpPrScheduler; DpPrDataManager; DpPrDataMap; PlDPR; PlDataGranule; GlUR;
S-DPS-20640	The PRONG CI shall be able to determine that a Special Data Product required for PGE execution requires staging.	PlDpr; DpPrScheduler; DpPrDataManager; DpPrDataMap; PlDPR; PlDataGranule; GlUR;

Table A-1. Requirements Trace (4 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-20650	The PRONG CI shall be able to determine that a Calibration Coefficient Data File required for PGE execution requires staging.	PlDpr; DpPrScheduler; DpPrDataManager; DpPrDataMap; PlDPR; PlDataGranule; GlUR;
S-DPS-20660	The PRONG CI shall be able to determine that a PGE requires staging.	PlDpr; DpPrScheduler; PlDPR; GlUR;
S-DPS-20670	The PRONG CI shall be able to determine that metadata associated with a PGE requires staging.	PlDpr; DpPrScheduler; PlDPR; GlUR;
S-DPS-20680	The PRONG CI shall support the movement of data from one Data Processing subsystem controlled storage resource to another Data Processing subsystem controlled storage resource.	DpPrDataManager; DpPrCotsManager; COTS; DpPrResourceManager; PlDPR; DpPrDataMap; PlDataGranule;
S-DPS-20690	The PRONG CI shall initiate the data staging process when the disk space required to support successful data staging is available.	DpPrDataManager; DsClESDTReferenceCollector; DsClRequest; DsClCommand; PlDPR; PlDataGranule; GlCallBack;
S-DPS-20700	The PRONG CI shall request data staging by sending a Data Request to the SDSRV CI .	DpPrDataManager; DsClESDTReferenceCollector; DsClRequest; DsClCommand; PlDPR; PlDataGranule; GlCallBack;
S-DPS-20710	The PRONG CI shall accept a Data Request Status message in response to the Data Request Message.	DpPrDataManager;
S-DPS-20720	The Data Request Status message shall inform the PRONG CI on the success or failure of data staging.	DpPrDataManager;
S-DPS-20730	The PRONG CI shall provide the capability to terminate the data staging process.	DpPrDataManager; GlCallBack; DpPrCotsManager; COTS;
S-DPS-20740	The PRONG CI shall send an Data Request message to the SDSRV CI to terminate the data staging process.	DpPrDataManager;
S-DPS-20750	The PRONG CI shall send a Complete Notification Status message to the source of the Data Processing Request if the data staging process was not completed successfully for the Data Processing Request.	DpPrDataManager;
S-DPS-20760	The Complete Notification Status message shall contain error information if the message was sent as a result of the failure of data staging.	DpPrDataManager;
S-DPS-20770	The PRONG CI shall accept ECS Data Products from the SDSRV CI.	DpPrDataManager;
S-DPS-20780	The PRONG CI shall accept metadata from the SDSRV CI.	DpPrDataManager;
S-DPS-20790	The PRONG CI shall accept PGEs from the SDSRV CI.	DpPrScheduler;
S-DPS-20800	The PRONG CI shall accept Calibration Coefficient data from the SDSRV CI.	DpPrDataManager;

Table A-1. Requirements Trace (5 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-20810	The PRONG CI shall accept Special Data Products from the SDSRV CI.	DpPrDataManager;
S-DPS-20820	The PRONG CI shall accept Ancillary Data Products from the SDSRV CI.	DpPrDataManager;
S-DPS-20830	The PRONG CI shall send a Data Insert Request message to the SDSRV CI CI to initiate the destaging of data.	DpPrCotsManager; COTS; DpPrDataManager; DpPrResourceManager; DsClESDTReference Collector;DsClRequest; DsClCommand; PlDPR; PlDataGranule; GlCallBack;
S-DPS-20840	The Data Request Status message shall inform the PRONG CI on the success or failure of data destaging.	DpPrDataManager; SDSRV CI;
S-DPS-20850	The PRONG CI shall destage Intermediate Data Products to the SDSRV CI.	DpPrCotsManager; COTS; DpPrDataManager; DpPrResourceManager; DsClESDTReference Collector;DsClRequest; DsClCommand; PlDPR; PlDataGranule; GlCallBack;
S-DPS-20860	The PRONG CI shall destage ECS Data Products to the SDSRV CI.	DpPrCotsManager; COTS; DpPrDataManager; DpPrResourceManager; DsClESDTReference Collector;DsClRequest; DsClCommand; PlDPR; PlDataGranule; GlCallBack;
S-DPS-20870	The PRONG CI shall send a Complete Notification Status message to the source of the Data Processing Request if the data destaging process was not completed successfully for the Data Processing Request.	DpPrDataManager;
S-DPS-20880	The Complete Notification Status message shall contain error information if the message was sent as a result of the failure of data destaging.	DpPrDataManager;

Table A-1. Requirements Trace (6 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-21000	The PRONG CI shall initiate execution of a PGE when the following is true: a. When all input data required to execute the PGE is available on local Data Processing subsystem storage resources. b. When the computer hardware resources are available to support execution of a PGE based on the computer hardware resource information associated with the Data Processing Request. c. When the Priority Information associated with the Data Processing Request has been fulfilled. d. When the maximum disk space requirements defined for the PGE are available to support the successful execution of the PGE. e. When the maximum memory resources defined for the PGE are available to support the successful execution of the PGE. f. When the CPU resources defined for the PGE are available to support the successful execution of the PGE.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPcf; DpPrPge; DpPrExecutionManager; DpPrResourceManager; PlDataGranule; PlDpr; PlDpr; DsClESDTReferenceCollector; DsClRequest; DsClCommand;
S-DPS-21070	The PRONG CI shall allocate disk space to support the execution of a PGE.	DpPrCotsManager; COTS; DpPrPge; DpPrExecutionManager; DpPrResourceManager; DpPrDataManager; MsMgCallBack; MsManager;
S-DPS-21080	The PRONG CI shall allocate memory to support the execution of a PGE.	DpPrCotsManager; COTS; DpPrPge; DpPrExecutionManager; DpPrResourceManager; DpPrDataManager; MsMgCallBack; MsManager;
S-DPS-21090	The PRONG CI shall allocate CPU to support the execution of a PGE.	DpPrCotsManager; COTS; DpPrPge; DpPrExecutionManager; DpPrResourceManager; DpPrDataManager; MsMgCallBack; MsManager;

Table A-1. Requirements Trace (7 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-21120	The PRONG CI shall create a Process Control File to provide information to the SDP Toolkit CI about the input data required to execute a PGE.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPcf; DpPrPge; DpPrExecutionManager; DpPrResourceManager; PlDataGranule; PlDpr; PlDpr; DsClESDTReferenceCollector; DsClRequest; DsClCommand;
S-DPS-21130	The PRONG CI shall create a Process Control File to provide information to the SDP Toolkit CI about the output data generated from the executing PGE.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPcf; DpPrPge; DpPrExecutionManager; DpPrResourceManager; PlDataGranule; PlDpr; PlDpr; DsClESDTReferenceCollector; DsClRequest; DsClCommand;
S-DPS-21140	The PRONG CI shall create a mapping of logical file handles to physical file handles in the Process Control File for the input data required to execute a PGE.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPcf; DpPrPge; DpPrExecutionManager; DpPrResourceManager; PlDataGranule; PlDpr; PlDpr; DsClESDTReferenceCollector; DsClRequest; DsClCommand;
S-DPS-21150	The PRONG CI shall create a mapping of logical file handles to physical file handles in the Process Control File for the output data generated from the executing PGE.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPcf; DpPrPge; DpPrExecutionManager; DpPrResourceManager; PlDataGranule; PlDpr; PlDpr; DsClESDTReferenceCollector; DsClRequest; DsClCommand;
S-DPS-21160	The PRONG CI shall create a Status Message File to be used by the SDP Toolkit CI to collect Toolkit status and error information about the execution of a PGE.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPge; DpPrExecutionManager; DpPrResourceManager; PlDpr;
S-DPS-21170	The PRONG CI shall create User Status Message Files to be used by the SDP Toolkit CI during PGE execution if requested through the data defining the characteristics of the PGE.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPge; DpPrExecutionManager; DpPrResourceManager; PlDpr;

Table A-1. Requirements Trace (8 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-21180	The PRONG CI shall allocate 1 shared memory attachment to a PGE to support access to internal memory during execution.	DpPrExecutionManager;
S-DPS-21210	The PRONG CI shall monitor the use of disk space by a PGE during execution.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPge; DpPrExecution-Manager; DpPrResource-Manager; MsMgCallBacks; MsManager; MsEvent;
S-DPS-21220	The PRONG CI shall take a predetermined error recovery action if the maximum disk space requirements defined for that PGE has been exceeded by an adaptable percentage value.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPge; DpPrExecution-Manager; DpPrResource-Manager; MsMgCallBacks; MsManager; MsEvent;
S-DPS-21230	The PRONG CI shall take a predetermined error recovery action if the maximum CPU time requirements defined for that PGE has been exceeded by an adaptable percentage value.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPge; DpPrExecution-Manager; DpPrResource-Manager; DpPrComputer; MsMgCallBacks; MsManager; MsEvent;
S-DPS-21240	The PRONG CI shall take a predetermined error recovery action if the maximum memory usage requirements defined for that PGE has been exceeded by an adaptable percentage value.	DpPrCotsManager; COTS; DpPrExecutables; DpPrPge; DpPrExecution-Manager; DpPrResource-Manager; DpPrComputer; MsMgCallBacks; MsManager; MsEvent;
S-DPS-21320	The PRONG CI shall use a SDP Toolkit API to associate Processing-Specific Metadata with each Granule of a generated Data Product.	DpPrExecutbale; DpPrPcf; DpPrPge; DpPrExecutionManger;
S-DPS-21330	The PRONG CI shall provide Processing-Specific Metadata to the SDP Toolkit to be associated with each Granule of a generated Data Product.	DpPrExecutbale; DpPrPcf; DpPrPge; DpPrExecutionManger;
S-DPS-21460	The PRONG CI shall use a SDP Toolkit API to associate Q/A-Specific Metadata with each Granule of a Data Product.	DpPrQaMonitor; PlDataTypes; PlDataType; AdCollection; Advertisement; DsClESDTReferenceCollector; GlParameter; GlParameterList; DsClCommand; DsClRequest;
S-DPS-21490	The PRONG CI shall record the Q/A-Specific Metadata of each input Data Product as part of the Q/A-Specific Metadata of the Granule of a Data Product.	DpPrExecutable; DpPrPcf; DpPrPge; DpPrExecutionManger;

Table A-1. Requirements Trace (9 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-21500	The PRONG CI shall use algorithms provided by the scientists to perform automated QA on generated Data Products.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots; DpPrDataManager; PlPge;
S-DPS-21510	The PRONG CI shall support the capability to update Q/A metadata as required by the execution of a PGE performing automated Q/A.	DpPrExecutable; DpPrPcf; DpPrPge; DpPrExecutionManger;
S-DPS-21520	The PRONG CI shall coordinate the deletion of the outputs of a PGE which were temporarily stored in the SDSRV CI.	DpPrCotsManager; Cots; DpPrDataManager; DpPrResourceManager; DsClESDTReferenceCollector; DsClRequest; DsClCommand; PlDPR; PlDataGranule; GlCallBack;
S-DPS-21530	The PRONG CI shall assign a unique Granule Identifier to each Granule of a generated Data Product.	DpPrDataManager; PlDataGranule;
S-DPS-21540	The PRONG CI shall destage all output data generated by a PGE to the SDSRV CI. (SEE Data Staging and Destaging Reqs for more details).	DpPrCotsManager; Cots; DpPrDataManager; DpPrResourceManager; DsClESDTReferenceCollector; DsClRequest; DsClCommand; PlDPR; PlDataGranule; GlCallBack;
S-DPS-21550	The PRONG CI shall not delete the output data generated by a PGE until the Data Request Status message is received from the SDSRV CI indicating that the output data was successfully copied to the SDSRV CI resources.	DpPrDataManager;
S-DPS-21560	If the resource fails during the execution of a PGE, the PRONG CI shall be capable of initiating the execution of the PGE without having to regenerate that PGE's input data.	DpPrCotsManager; Cots; DpPrExecutable; DpPrPcf; DpPrPge; DpPrExecutionManager; DpPrResourceManager; DpPrDataManager; MsMgCallBacks; MsManager; MsEvent;
S-DPS-21570	If a PGE fails abnormally during execution, the PRONG CI shall be capable of initiating the execution of the PGE without having to regenerate that PGE's input data.	DpPrCotsManager; Cots; DpPrExecutable; DpPrPcf; DpPrPge; DpPrExecutionManager; DpPrResourceManager; DpPrDataManager; PlDataGranule; PlDpr; MsEvent; DsClESDTReferenceCollector; DsClRequest; DsClCommand;

Table A-1. Requirements Trace (10 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-21580	The PRONG CI shall send a Complete Notification Status message to the source of the Data Processing Request at the completion of PGE execution if the execution was terminated by the PRONG CI or the outputs of the PGE did not require destaging.	DpPrCotsManager; Cots; DpPrPge; DpPrExecutionManager; DpPrResourceManager; DpPrDataManager;
S-DPS-21590	Upon the completion of destaging, the PRONG CI shall send a Complete Notification Status message to the source of the Data Processing Request.	DpPrCotsManager; Cots; DpPrPge; DpPrExecutionManager; DpPrResourceManager; DpPrDataManager;
S-DPS-21700	The operations staff shall have the capability of terminating the data staging process for a Data Processing Request.	DpPrDataManager; DsClESDTReferenceCollector; DsClRequest; DsClCommand; PlDataGranule; PlDpr; GlCallBack;
S-DPS-21710	The operations staff shall have the capability of terminating the data destaging process for a Data Processing Request.	DpPrCotsManager; Cots; DpPrDataManager; DpPrResourceManager; DsClESDTReferenceCollector; DsClRequest; DsClCommand; PlDataGranule; PlDpr; GlCallBack;
S-DPS-21720	The operations staff shall have the capability of canceling the processing of a Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots; DpPrDataManager; DpPrExecutionManager;
S-DPS-21750	The operations staff shall have the capability of modifying the information associated with the Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-21760	The operations staff shall have the capability of viewing the Processing Queues.	DpPrCotsManager; Cots;
S-DPS-21770	The operations staff shall have the capability of requesting the status of a Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-21780	The operations staff shall have the capability of reporting resource management information.	DpPrCotsManager; Cots; DpPrResourceManager; DpPrExecutionManager; DpPrComputer; DpPrDiskPartition;
S-DPS-21790	The operations staff shall have the capability of viewing a Data Product.	DpPrQaMonitor; DsClESDTReference; PlDataGranules; PlDataGranule; EOSVIEW;
S-DPS-21800	The operations staff shall have the capability of viewing the algorithms used to generate a Data Product.	DpPrQaMonitor;
S-DPS-21810	The operations staff shall have the capability of viewing the ECS Data Products used to generate a Data Product.	DpPrQaMonitor;

Table A-1. Requirements Trace (11 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-21820	The operations staff shall have the capability of viewing the Calibration Coefficient Data used to generate a Data Product.	DpPrQaMonitor;
S-DPS-21830	The operations staff shall have the capability of viewing the Ancillary Data Products used to generate a Data Product.	DpPrQaMonitor;
S-DPS-21840	The operations staff shall have the capability of viewing the Status Information files associated with the generated Data Product.	COTS;
S-DPS-21850	The operations staff shall have the capability of viewing all metadata associated with the generation of a Data Product.	DpPrQaMonitor;
S-DPS-21880	The PRONG CI shall provide a User Interface to authorized users.	Cots; DpPrScheduler; DpPrDataManager; DpPrExecutionManager; DpPrResourceManager;
S-DPS-21890	The PRONG CI shall provide a Processing Queue Display as a visual display of the Processing Queues.	DpPrCotsManager; Cots;
S-DPS-21900	The PRONG CI shall update the Processing Queue Display information when the Processing State of a queued Data Processing Request is modified.	DpPrCotsManager; Cots;
S-DPS-21910	The PRONG CI shall update the Processing Queue Display information with an alert message when a fault has occurred during the queue processing of a Data Processing Request.	DpPrCotsManager; Cots;
S-DPS-21920	The PRONG CI shall update the Processing Queue Display information with an alert message when a fault has occurred during the data staging process.	DpPrDataManager;
S-DPS-21930	The PRONG CI shall update the Processing Queue Display information with an alert message when a fault has occurred during the execution of a PGE.	DpPrExecutionManager;
S-DPS-21940	The PRONG CI shall update the Processing Queue Display information with an alert message when a fault has occurred during the data destaging process.	DpPrDataManager;
S-DPS-21950	The PRONG CI shall log all alert messages which are used to update the Processing Queue display information.	COTS;
S-DPS-21960	The PRONG CI shall provide a user interface to cancel the processing of a Data Processing Request.	DpPrScheduler;
S-DPS-21970	The PRONG CI shall provide a user interface to modify the Priority Information associated with a Data Processing Request.	DpPrScheduler;
S-DPS-21980	The PRONG CI shall provide a user interface to modify the information associated with a Data Processing Request.	DpPrScheduler;
S-DPS-21990	The PRONG CI shall provide a user interface to suspend the processing of a Data Processing Request.	DpPrScheduler;
S-DPS-22000	The PRONG CI shall provide a user interface to resume suspended processing of a Data Processing Request.	DpPrScheduler;
S-DPS-22010	The PRONG CI shall provide a user interface to view the data associated with the Data Processing Request.	DpPrScheduler;

Table A-1. Requirements Trace (12 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-22020	The PRONG CI shall provide a user interface to support the manual Q/A of Data Products.	DpPrQaMonitor;
S-DPS-22030	The PRONG CI shall provide access to data visualization tools to support the manual Q/A of Data Products.	DpPrQaMonitor; DsClES-DTReferenc; PlDataGranules; PlDataGranule; EOSVIEW;
S-DPS-22040	The PRONG CI shall provide a user interface to support the update of the Q/A metadata of a Data Product.	DpPrQaMonitor;
S-DPS-22050	The PRONG CI shall provide an interface to support the visual display of a Data Product.	DpPrQaMonitor; EOS-VIEW;
S-DPS-22060	The PRONG CI shall provide an interface to support the visual display of the algorithms used to generate a Data Product.	DpPrQaMonitor;
S-DPS-22070	The PRONG CI shall provide an interface to support the visual display of the ECS Data Products used to generate a Data Product.	DpPrQaMonitor;
S-DPS-22080	The PRONG CI shall provide an interface to support the visual display of the Calibration Coefficient Data used to generate a Data Product.	DpPrQaMonitor;
S-DPS-22090	The PRONG CI shall provide an interface to support the visual display of the Ancillary Data Products used to generate a Data Product.	DpPrQaMonitor;
S-DPS-22100	The PRONG CI shall provide an interface to support the visual display of the Status Information files associated with the generated Data Product.	COTS; DpPrQaMonitor;
S-DPS-22110	The PRONG CI shall provide an interface to support the visual display of all metadata associated with the generation of a Data Product.	DpPrQaMonitor;
S-DPS-22120	The PRONG CI shall support a capability to alert the operations staff of a Data Product which is being stored temporarily in the Data Server.	DpPrDataManager;
S-DPS-22130	The PRONG CI shall support a capability to alert the operations staff of a Data Product which requires quality assurance activities.	DpPrQaMonitor;
S-DPS-22200	The PRONG CI shall accept a Processing Information Request to request the status of a Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22210	The PRONG CI shall have the capability to provide status for a Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22220	The PRONG CI shall provide current DPR Processing State data as part of the status information of a Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22230	The PRONG CI shall provide current queue position as part of the status information of a Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22240	The PRONG CI shall provide status information for the PGE associated with the Data Processing Request if the PGE is currently executing.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;

Table A-1. Requirements Trace (13 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-22250	The PRONG CI shall have the capability of receiving the Status Information File of an executing PGE from the Data Processing Subsystem resource executing the PGE.	DpPrDataManager; COts;
S-DPS-22400	The PRONG CI shall accept Operations Commands to suspend, resume, or cancel the processing of a Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22410	The PRONG CI shall accept an Operations Command to modify a Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22470	The PRONG CI shall update the DPR Processing State to cancel when the Operation Command specifies cancellation.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22480	The PRONG CI shall terminate data staging if in progress when the Data Processing Request is canceled.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22490	The PRONG CI shall deallocate the memory which was allocated to the executing PGE associated with the canceled Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22500	The PRONG CI shall deallocate the disk storage which was allocated to the executing PGE associated with the canceled Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22510	The PRONG CI shall deallocate the CPU which was allocated to the executing PGE associated with the canceled Data Processing Request.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22520	The PRONG CI shall terminate the execution of the PGE if in progress when the Data Processing Request is canceled.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22530	The PRONG CI shall terminate data destaging if in progress when the Data Processing is canceled.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22540	The PRONG CI shall send a Complete Notification Status message to the source of the Data Processing Request when the Data Processing Request is canceled.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22620	The PRONG CI shall update the Priority Information associated with the Data Processing Request with the Priority Information contained in the Operation Command which specifies modify.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-22630	The PRONG CI shall perform queue processing for a Data Processing Request which has updated Priority Information.	PlDpr; DpPrScheduler; DpPrCotsManager; Cots;
S-DPS-30610	The DPREP CI shall assess the quality of onboard attitude data contained in the TRMM spacecraft ancillary data by detecting and noting in metadata for: a) missing data b) erroneous data (i.e., invalid Euler angle, invalid Euler angle rate)	See Note 1; DpPpTrmmOnBoardAttitudeData; DpPpCheckAttitudeQuality;
S-DPS-30700	The DPREP CI shall provide to the SDP Toolkit, at a minimum, the following metadata with the ephemeris data files for TRMM processing: a. Time range, b. Orbit number range, c. Platform.	See Note 1; DpPpTrmmOnBoardAttitudeData;

Table A-1. Requirements Trace (14 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-30720	The DPREP CI shall provide, at a minimum, the following ephemeris data files to the SDP Toolkit for TRMM processing: a. Platform position velocity vectors b. Platform attitude/attitude rate data (expressed as Euler angles in radians and radians/s, respectively)	See Note 1; DpPpTrmmOnBoardAttitudeData; DpPpGetAttitudePacket;DpPpFdfTrmmDefinitiveOrbitData; DpPpGetEphemRecord;
S-DPS-30740	The DPREP CI shall provide to the SDP Toolkit orbit and attitude data in the native format of the host hardware for TRMM processing.	See Note 1; DpPpTrmmOnBoardAttitudeData; DpPpGetAttitudePacket; DpPpFdfTrmmDefinitiveOrbitData; DpPpGetEphemRecord;
S-DPS-30760	The DPREP CI shall provide to the SDP Toolkit orbit and attitude data in HDF-EOS format for TRMM processing.	See Note 1; DpPpTrmmOnBoardAttitudeData; DpPpGetAttitudePacket; DpPpArchiveAttitudeData; DpPpFdfTrmmDefinitiveOrbitData; DpPpGetEphemRecord; DpPpArchiveEphemerisData;
S-DPS-30800	The DPREP CI shall provide to the SDP Toolkit SDPF-generated L0 production data files with unique APIDs as defined in the SDPF-ECS ICD.	See Note 1; DPPpPreprocessingData; DpPpLevelZeroData; DpPpSdpfLevelZeroProductionData;
S-DPS-30810	The DPREP CI shall provide to the SDP Toolkit SDPF-generated L0 production data files as two separate files: (a) A Standard Format Data Unit (SFDU) header file, (b) Data Set File as defined in the SDPF-ECS ICD.	See Note 1; DPPpPreprocessingData; DpPpLevelZeroData; DpPpSdpfLevelZeroProductionData; DpPpSdpfLevelZeroDatasetFile; DpPpSdpfLevelZeroSfduFile;
S-DPS-31020	The DPREP CI shall provide, at a minimum, the following metadata information to the SDP Toolkit with SDPF-generated L0 data: a. Actual start time of staged L0 data b. Actual end time of staged L0 data c. Number of physical L0 data files staged d. Start time of L0 data as requested by EOS investigators through the planning/processing system e. End time of L0 data as requested by EOS investigators through the planning/processing system f. APID of each L0 data file, of the L0 data files are APID-unique g. Orbit number of the staged L0 data file	See Note 1; DPPpPreprocessingData; DpPpLevelZeroData; DpPpSdpfLevelZeroProductionData; DpPpSdpfLevelZeroDatasetFile; DpPpSdpfLevelZeroSfduFile;

Table A-1. Requirements Trace (15 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-31620	The DPREP CI shall maintain and prepare on an ad hoc basis, at a minimum, the following GFE static data sets for input to the SDP Toolkit: a. Digital terrain map data sets b. Land/Sea data sets c. Digital political map data sets	See Note 1; DpPrDataManager;
S-DPS-31700	The DPREP CI shall extract metadata attributes for TBD FDF and external Ancillary Data sets, in addition to metadata extraction by the INGST CI.	See Note 1; DpPpPreprocessingData;
S-DPS-40010	The AITTL CI shall have the capability to receive a Science Software Delivery from the SCF electronically via the network.	INGST CSCI;
S-DPS-40020	The AITTL CI shall have the capability to receive a Science Software Delivery from the Science Data Server.	SDSRV CSCI;
S-DPS-40030	The AITTL CI shall provide the operations staff with the capability to register a Subscription with the Data Server to be notified when a new Science Software Delivery is received.	SDSRV CSCI;
S-DPS-40040	The AITTL CI shall provide the operations staff with the capability to request transfer of the Science Software Delivery files from the Data Server to the local I&T area.	SDSRV CSCI;
S-DPS-40100	The AITTL CI shall provide the operations staff with the capability to display Science Software documentation stored in any of the following formats: a) PostScript, b) ASCII, c) Hypertext Markup Language (HTML), d) Microsoft Word, e) WordPerfect, f) Adobe Acrobat Portable Document Format (PDF).	Documentation Viewing Tools;
S-DPS-40110	The AITTL CI shall provide the operations staff with the capability to print Science Software documentation stored in any of the following formats: a) PostScript, b) ASCII, c) Hypertext Markup Language (HTML), d) Microsoft Word, e) WordPerfect, f) Adobe Acrobat Portable Document Format (PDF).	Documentation Viewing Tools;
S-DPS-40200	The AITTL CI shall have the capability to verify that Science Software source code written in C complies with the ANSI standard specification for C.	AITHW CI;
S-DPS-40210	The AITTL CI shall have the capability to verify that Science Software source code written in FORTRAN77 complies with the ANSI standard specification for FORTRAN77.	AITHW CI;
S-DPS-40230	The AITTL CI shall have the capability to verify that Science Software source code written in FORTRAN 90 complies with the ANSI standard specification for FORTRAN 90.	AITHW CI;
S-DPS-40250	The AITTL CI shall have the capability to verify that Science Software source code written in Ada complies with the military specification MIL-STD-1815-A.	AITHW CI;
S-DPS-40260	The AITTL CI shall have the capability to verify that Science Software source code is POSIX-compliant.	Standards Checker Tools; Static and Dynamic Code Checker Tools;

Table A-1. Requirements Trace (16 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-40280	The AITTL CI shall have the capability to verify that Science Software source code and Science Software scripts follow the following SDP Toolkit usage requirements (from 194-809-SD4-001, PGS Toolkit Users Guide for the ECS Project):	Standards Checker Tools; Static and Dynamic Code Checker Tools;
S-DPS-40295	The AITTL CI shall provide standards checking capabilities, including, but not limited to: a. Flagging whenever a bit operation is used on signed numbers. (C only) b. Flagging argument list mismatches (type and number of arguments).	Standards Checker Tools; Static and Dynamic Code Checker Tools;
S-DPS-40320	The AITTL CI shall have the capability to verify that Science Software source code includes headers as specified in 423-16-01, Data Production Software and Science Computing Facility (SCF) Standards and Guidelines.	Standards Checker Tools; Static and Dynamic Code Checker Tools;
S-DPS-40340	The AITTL CI shall have the capability to generate report files describing the results of standards checking.	Report Generation Tools;
S-DPS-40400	The AITTL CI shall have the capability to determine if the Science Software contains memory leaks.	AITHW CI; Static and Dynamic Code Checker Tools;
S-DPS-40405	The AITTL CI shall have the capability to determine if the Science Software contains out of bounds indexing.	AITHW CI; Static and Dynamic Code Checker Tools;
S-DPS-40430	The AITTL CI shall have the capability to generate report files describing the results of code analysis.	Report Generation Tools;
S-DPS-40700	The data visualization capability of the AITTL CI shall include the capability to display data in hexadecimal, octal, decimal, or ASCII form.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40710	The data visualization capability of the AITTL CI shall include the capability to display data as a two- or three-dimensional image.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40720	The data visualization capability of the AITTL CI shall include the capability to display data as a two- or three-dimensional plot.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40730	The data visualization capability of the AITTL CI shall include the capability to difference data and to display the differences as a two- or three-dimensional image or plot.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40740	The data visualization capability of the AITTL CI shall include the capability to produce and play a "movie loop" of data in two- or three-dimensional image or plot form.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40750	The data visualization capability of the AITTL CI shall include the capability to display an arbitrary two-dimensional slice of a three-dimensional image or plot.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40760	The data visualization capability of the AITTL CI shall include the capability to rotate a three-dimensional image or plot about an arbitrary axis.	Data Visualization Tools; ECS HDF Visualization Tools

Table A-1. Requirements Trace (17 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-40770	The data visualization capability of the AITTL CI shall include providing the user with the option to specify the color table for new or existing image displays.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40780	The data visualization capability of the AITTL CI shall include providing the user with the option to specify the axis limits for new or existing plot displays.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40790	The data visualization capability of the AITTL CI shall include providing the operations staff with the option to specify the parameter assigned to each axis in new or existing plot or image displays.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40800	The data visualization capability of the AITTL CI shall include the capability to display simultaneously multiple views of the same or different data in different windows.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40810	The data visualization capability of the AITTL CI shall include the capability to save any plot, image, or hex/decimal/octal/ ASCII dump to a file.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40820	The data visualization capability of the AITTL CI shall include feature enhancement capabilities, including but not limited to (1) histogram equalization and (2) edge enhancement.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40830	The data visualization capability of the AITTL CI shall include the capability to read ASCII, binary, or HDF files.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40840	The data visualization capability of the AITTL CI shall include the capability to allow the operations staff to specify a custom input data format.	Data Visualization Tools; ECS HDF Visualization Tools
S-DPS-40900	The AITTL CI shall have the capability to find all differences between two data files which are greater than some specified absolute threshold.	File Comparison Utility Tools;
S-DPS-40910	The AITTL CI shall have the capability to find all differences between two data files which are greater than some specified relative threshold.	File Comparison Utility Tools;
S-DPS-40920	The AITTL CI shall have the capability to generate report files describing the results of file comparisons.	File Comparison Utility Tools; Report Generation Tools;
S-DPS-40930	The file comparison capability of the AITTL CI shall include the capability to read ASCII, binary, or HDF files.	File Comparison Utility Tools;
S-DPS-40940	The file comparison capability of the AITTL CI shall include the capability to allow the operations staff to specify a custom data format.	File Comparison Utility Tools;
S-DPS-41000	The AITTL CI shall have the capability to measure the CPU time of a process.	AITHW CI;
S-DPS-41005	The AITTL CI shall have the capability to measure the wall clock time of a process.	AITHW CI;
S-DPS-41010	The AITTL CI shall have the capability to measure the CPU time of each procedure within a process.	AITHW CI;
S-DPS-41015	The AITTL CI shall have the capability to measure the wall clock time of each procedure within a process.	AITHW CI;

Table A-1. Requirements Trace (18 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-41020	The AITTL CI shall have the capability to measure the memory usage of a process.	AITHW CI;
S-DPS-41030	The AITTL CI shall have the capability to measure the disk space usage of a process.	AITHW CI;
S-DPS-41035	The AITTL CI shall have the capability to count the number of page faults for a process.	Profiling Tools;
S-DPS-41040	The AITTL CI shall have the capability to count the number of I/O accesses made by a process to each of its input and output data files.	Profiling Tools;
S-DPS-41050	The AITTL CI shall have the capability to generate report files discussing the results of profiling activities.	AITHW CI; Profiling Tools; Report Generation Tools;
S-DPS-41300	The AITTL CI shall provide to the operations staff, via a GUI, the capability to display a list of PGE Database Entries.	PGE Database Update GUI Tool; PLANG CI;
S-DPS-41310	The AITTL CI shall provide to the operations staff, via a GUI, the capability to display a specific PGE Database Entry.	PGE Database Update GUI Tool; PLANG CI;
S-DPS-41320	The AITTL CI shall provide to the operations staff, via a GUI, the capability to modify a specific PGE Database Entry.	PGE Database Update GUI Tool; PLANG CI;
S-DPS-41330	The AITTL CI shall provide to the operations staff, via a GUI, the capability to add a new PGE Database Entry.	PGE Database Update GUI Tool; PLANG CI;
S-DPS-41340	The AITTL CI shall provide to the operations staff, via a GUI, the capability to remove a specific PGE Database Entry.	PGE Database Update GUI Tool; PLANG CI;
S-DPS-41350	The AITTL CI shall provide to the operations staff, via a GUI, cut, copy, and paste capability for a PGE Database Entry.	PGE Database Update GUI Tool; PLANG CI;
S-DPS-41400	The DAAC I&T environment shall include access to a configuration management tool supplied by MSS.	MSS CI;
S-DPS-41410	The DAAC I&T environment shall include access to a problem tracking tool supplied by MSS.	MSS CI;
S-DPS-41500	The AITTL CI shall provide the capability for operations staff to write reports. This capability will include: (a) word processing, (b) spreadsheet, (c) plotting, (d) drawing.	Report Generation Tools;
S-DPS-41510	The AITTL CI shall provide templates for reports to be written by the operations staff. (NOTE: It is assumed that these templates will be developed by the Science Office.)	Report Generation Tools;
S-DPS-41520	The AITTL CI shall provide the capability for operations staff to keep a running log of integration and test activities on-line.	Report Generation Tools;
S-DPS-41530	The AITTL CI shall provide the capability for authorized users to examine the integration and test logs and other reports.	Report Generation Tools;
S-DPS-41895	The AITTL CI shall provide to the operations staff the capability to retrieve a specified data file from local DAAC storage.	AITHW CI; ???
S-DPS-41900	The AITTL CI shall provide to the operations staff, via a GUI, the capability to retrieve a specified data file from a specified Data Server.	Manual Staging GUI Tool;
S-DPS-42000	The AITTL CI shall provide the operations staff with the capability to view the metadata associated with a data file.	Product Metadata Display Tool;
S-DPS-42005	The AITTL CI shall provide the operations staff with the capability to edit the metadata associated with a data file.	SDSRV CI;

Table A-1. Requirements Trace (19 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-42010	The AITTL CI shall provide the operations staff with the capability to write the metadata associated with a data file to a report file.	Product Metadata Display Tool; Report Generation Tools;
S-DPS-42100	The operations staff shall place a Science Software Delivery Package in a non-public directory accessible to the hardware scheduled to be used for I&T.	Operational;
S-DPS-42110	The operations staff shall read and/or review all documentation included in the Delivery Package.	Operational;
S-DPS-42120	The operations staff shall perform automated checking of all source code included in the Delivery Package against established coding standards and guidelines.	Operational;
S-DPS-42130	The operations staff shall perform automated checking of all scripts included in the Delivery Package against established coding standards and guidelines.	Operational;
S-DPS-42140	The operations staff shall have the capability to perform static analyses of source code for (at a minimum) argument mismatches and variables set before used.	Operational;
S-DPS-42150	The operations staff shall have the capability to examine all test data and expected test results files included in the Delivery Package to verify completeness and correct format.	Operational;
S-DPS-42160	The operations staff shall have the capability to examine all coefficient files included in the Delivery Package to verify completeness and correct format.	Operational;
S-DPS-42170	The operations staff shall have the capability to compile all FORTRAN77, FORTRAN 90 and C source code included in the Delivery Package.	Operational;
S-DPS-42175	The operations staff shall have the capability to compile all Ada source code included in the Delivery Package for CERES.	Operational;
S-DPS-42180	The operations staff shall check source code, coefficient files, test plans, test data, expected test results and other documentation into the Configuration Management tool.	Operational;
S-DPS-42190	The operations staff (and others who are specifically authorized) shall have the capability to check out source code, coefficient files, test plans, test data, expected test results and other documentation from the Configuration Management tool.	Operational;
S-DPS-42200	Whenever a Science Software Delivery is received by the AITTL CI directly from the SCF via the network, the operations staff shall notify the SCF that the delivery has been received successfully.	Operational;
S-DPS-42300	The operations staff shall have the capability to link FORTRAN77, FORTRAN 90 and C object code with the SCF version of the SDP Toolkit.	Operational;
S-DPS-42305	The operations staff shall have the capability to link Ada object code for CERES with the SCF version of the SDP Toolkit.	Operational;

Table A-1. Requirements Trace (20 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-42310	The operations staff shall link FORTRAN77, FORTRAN 90 and C object code with the DAAC version of the SDP Toolkit.	Operational;
S-DPS-42315	The operations staff shall link Ada object code for CERES with the DAAC version of the SDP Toolkit.	Operational;
S-DPS-42320	The operations staff shall have the capability to link FORTRAN77, FORTRAN 90 and C object code with other libraries.	Operational;
S-DPS-42325	The operations staff shall have the capability to link Ada object code for CERES with other libraries.	Operational;
S-DPS-42330	The operations staff shall have the capability to run binary executables without impacting other ongoing DAAC activities.	Operational;
S-DPS-42340	The operations staff shall have the capability to perform dynamic analyses of source code for (at a minimum) memory leaks, out of bounds indexing, and distribution of resource demands.	Operational;
S-DPS-42350	The operations staff shall have the capability to execute perl, C shell or Bourne shell scripts.	Operational;
S-DPS-42360	The operations staff shall have the capability of determining the computing resources utilized by an execution of a PGE; viz., PGE CPU time, system CPU time, elapsed time, percent elapsed time, shared memory use, maximum memory used, number of page faults, number of swaps, number of block input operations, and number of block output operations.	Operational;
S-DPS-42370	The operations staff shall collect during I&T the performance and resource utilization information needed for entry into or update of the PGE data base.	Operational;
S-DPS-42500	The operations staff shall execute the Test Plans included in the Delivery Package.	Operational;
S-DPS-42510	The operations staff shall have the capability of displaying Data Products.	Operational;
S-DPS-42520	The operations staff shall have the capability of displaying data in intermediate files used to generate a Data Product.	Operational;
S-DPS-42530	The operations staff shall have the capability of displaying data in input files used to generate a Data Product.	Operational;
S-DPS-42540	The operations staff shall have the capability of displaying data in coefficient files used to generate a Data Product.	Operational;
S-DPS-42550	The operations staff shall have the capability of displaying the Ancillary Data used to generate a Data Product .	Operational;
S-DPS-42560	The operations staff shall have the capability of viewing the Status Information files associated with the generated Data Product.	Operational;
S-DPS-42570	The operations staff shall have the capability of displaying all metadata associated with the generation of a Data Product.	Operational;
S-DPS-42580	The operations staff shall have the capability of comparing data in two coefficient files.	Operational;

Table A-1. Requirements Trace (21 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-42590	The operations staff shall have the capability of comparing two Data Product files.	Operational;
S-DPS-42600	The operations staff shall have the capability of comparing data in two intermediate files.	Operational;
S-DPS-42610	The operations staff shall enter new PGEs into the PGE Database, along with their performance and resource utilization information.	Operational;
S-DPS-42620	The operations staff shall update information the PGE Database as necessary to reflect changes in performance and resource utilization resulting from a modification to a PGE.	Operational;
S-DPS-42630	The operations staff shall have the capability of run PGEs in a parallel test or for a commissioning period, utilizing the Planning and Processing Subsystems and the Product output flagged as "test".	Operational;
S-DPS-42640	The operations staff shall have the capability to send the test results to the SCF for analysis.	Operational;
S-DPS-42650	The operations staff shall have the capability to write ad hoc test tools using the perl, C shell or Bourne shell script languages.	Operational;
S-DPS-42660	The operations staff shall have the capability to write ad hoc test tools using the FORTRAN77, FORTRAN 90 and C programming languages.	Operational;
S-DPS-42700	The operations staff shall have the capability to enter and track discrepancy reports related to AI&T.	Operational;
S-DPS-42710	The operations staff shall have the capability to send to and receive email messages from Science Software Developer staff and ECS staff.	Operational;
S-DPS-42720	The operations staff shall have the capability to engage in teleconferences with Science Software Developer staff and ECS staff.	Operational;
S-DPS-42740	The operations staff shall reports on the status of I&T-related discrepancy reports.	Operational;
S-DPS-42750	The operations staff shall have the capability of record each step performed during I&T, the results and actions initiated, if any.	Operational;
S-DPS-42760	The operations staff shall report on the status of the I&T activities each PGE.	Operational;
S-DPS-42770	The operations staff shall have the capability of writing an Inspection Report for each Science Software Delivery.	Operational;
S-DPS-42780	The operations staff shall have the capability of writing an Integration Report for each Science Software Delivery.	Operational;
S-DPS-42790	The operations staff shall have the capability of writing an Acceptance Test Report for each Science Software Delivery.	Operational;
S-DPS-60010	The SPRHW CI shall support the capability to manage, queue, and execute processes on the processing resources at each DAAC site.	SPRHW CI;

Table A-1. Requirements Trace (22 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-60020	The SPRHW CI shall support the capability to stage and destage data.	SPRHW CI;
S-DPS-60050	The SPRHW CI shall contain and/or provide access to staging (working storage), I/O and processing resources necessary to perform routine processing.	SPRHW CI;
S-DPS-60060	The SPRHW CI shall have a Fail-Soft capability to meet RMA requirements.	SPRHW CI;
S-DPS-60080	The SPRHW CI shall have provision for Initialization, Recovery, and an orderly shutdown.	SPRHW CI;
S-DPS-60090	The SPRHW CI shall support startup and initialization to be completed within 30 minutes (TBR)	SPRHW CI;
S-DPS-60100	The SPRHW CI shall support shutdown to be completed within 30 minutes (TBR).	SPRHW CI;
S-DPS-60110	The SPRHW CI shall have a fault detection/fault isolation capability of major HWCI component failures without interfering with operations.	SPRHW CI;
S-DPS-60120	The SPRHW CI shall have a status monitoring capability.	SPRHW CI;
S-DPS-60135	The SPRHW CI design and implementation shall have the flexibility to accommodate Science Processing expansion up to a factor of 3 in its capacity with no changes in its design and up to a factor of 10 without major changes to its design.	SPRHW CI;
S-DPS-60160	The SPRHW CI shall support collection and maintenance for Fault Management, configuration, performance, accountability, and security of Processing CI hardware resources.	SPRHW CI;
S-DPS-60230	The SPRHW CI shall provide a phased capacity to support: a. for pre-launch AI&T at launch minus 2 years: 0.3 X, where X is defined as the at-launch processing estimate b. for pre-launch AI&T and System I&T at-launch minus 1 year: 1.2 X, where X is defined as the at-launch processing estimate c. for post-launch AIT, standard processing, and reprocessing, starting at launch plus 1 year: 2.2 X, where X is defined as the standard processing estimate for that period d. for post-launch AIT, standard processing, and reprocessing, starting at launch plus 2 years: 4.2 X, where X is defined as the standard processing estimate for that period.	SPRHW CI;
S-DPS-60240	The SPRHW CI shall support a total processing requirement as derived from Table 1 (Appendix E Section E.1).	SPRHW CI;
S-DPS-60250	The SPRHW CI shall be able to support a data volume (GB/Day) as derived from Table 1 (Appendix E Section E.1).	SPRHW CI;
S-DPS-60330	The SPRHW CI shall have the capacity to support I/O to temporary and intermediate storage or multiple passes over input Products as required by individual science software.	SPRHW CI;

Table A-1. Requirements Trace (23 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-60350	The SPRHW CI shall generate Level 1 Standard Products within 24 hours after processing is initiated.	SPRHW CI;
S-DPS-60360	The SPRHW CI shall generate Level 2 Standard Products within 24 hours after processing is initiated.	SPRHW CI;
S-DPS-60370	The SPRHW CI shall generate Level 3 Standard Products within 24 hours after processing is initiated.	SPRHW CI;
S-DPS-60380	The SPRHW CI shall generate and make available to the users Level 4 Standard Products within one week after the availability to ECS of all necessary Level 3 and other input data sets.	SPRHW CI;
S-DPS-60450	Each computer providing product generation capability shall have an operational availability of 0.95 at a minimum.	SPRHW CI;
S-DPS-60480	The SPRHW CI shall have provision for the AIT science processor to be a backup to the production science processor in the event of a failure.	SPRHW CI;
S-DPS-60490	The SPRHW CI shall be capable of supporting system development without impact to normal operations.	SPRHW CI;
S-DPS-60500	The SPRHW CI shall be capable of supporting science software test without impact to normal operations.	SPRHW CI;
S-DPS-60510	The SPRHW CI shall be capable of supporting system upgrades while meeting specified operational availability requirements.	SPRHW CI;
S-DPS-60520	The SPRHW CI elements and components shall include the on-line (operational mode) and off-line (test mode) fault detection and isolation capabilities required to achieve the specified operational availability requirements.	SPRHW CI;
S-DPS-60610	The SPRHW CI platforms shall have provision for interfacing with one or more Local Area Networks (LANs).	SPRHW CI;
S-DPS-60610	The SPRHW CI platforms shall have provision for interfacing with one or more Local Area Networks (LANs).	SPRHW CI;
S-DPS-60612	The SPRHW CI platforms shall have provision for interfacing with Data Server.	SPRHW CI;
S-DPS-60615	The SPRHW CI platforms shall have provision for interfacing with Ingest	SPRHW CI;
S-DPS-60617	The SPRHW CI platforms shall have provision for interfacing with Planning.	SPRHW CI;
S-DPS-60710	The electrical power requirements for SPRHW CI equipment shall be in accordance with ECS Facilities Plan (DID 302/DV2)	SPRHW CI;
S-DPS-60740	The air conditioning requirements for the SPRHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	SPRHW CI;
S-DPS-60750	The grounding requirements for SPRHW CI equipment shall be in accordance with ECS Facilities Plan (DID 302/DV2).	SPRHW CI;
S-DPS-60760	The fire alarm requirements for SPRHW CI equipment shall be in accordance with ECS Facilities Plan (DID 302/DV2).	SPRHW CI;

Table A-1. Requirements Trace (24 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-60770	The acoustical requirements for SPRHW CI equipment shall be in accordance with ECS Facilities Plan (DID 302/DV2).	SPRHW CI;
S-DPS-60780	The physical interface requirements between SPRHW CI equipment and the facility shall be in accordance with ECS Facilities Plan (DID 302/DV2).	SPRHW CI;
S-DPS-60790	The footprint size and the physical layout of SPRHW CI equipment shall be in accordance with the and ECS Facilities Plan (DID 302/DV2).	SPRHW CI;
S-DPS-60910	The SPRHW CI shall support test activities throughout the development phase.	SPRHW CI;
S-DPS-60920	The following testing shall be performed on the SPRHW CI:	SPRHW CI;
S-DPS-60930	The SPRHW CI shall provide test tools as designated in the SDPS Test Tool Matrix.	SPRHW CI;
S-DPS-60940	The SPRHW CI shall be capable of simultaneously supporting the Independent Verification & Validation (IV&V) activities and the ECS development activities, both before and after flight operations begin.	SPRHW CI;
S-DPS-60950	The SPRHW CI shall be capable of supporting end-to-end test and verification activities of the EOS program including during the pre-launch, spacecraft verification, and instrument verification phases.	SPRHW CI;
S-DPS-60960	The SPRHW CI shall support end-to-end EOS system testing and fault isolation.	SPRHW CI;
S-DPS-60970	The SPRHW CI shall be capable of being monitored during testing.	SPRHW CI;
S-DPS-61040	The SPRHW CI computer platform shall provide a hard media device with a capacity of TBD GB for software and system maintenance and upgrade support.	SPRHW CI;
S-DPS-61045	The SPRHW CI shall provide local consoles for maintenance and operation.	SPRHW CI;
S-DPS-61110	The operating system for each Unix platform in the SPRWHW CI shall conform to the POSIX.2 standard.	SPRHW CI;
S-DPS-61120	The SPRHW CI POSIX.2 compliant platform shall have the following utilities installed at a minimum: perl, emacs, gzip, tar, imake, prof, gprof, nm.	SPRHW CI;
S-DPS-61130	The SPRHW CI POSIX.2 compliant platform shall have the following POSIX.2 user Portability Utilities installed at a minimum: man, vi.	SPRHW CI;
S-DPS-61140	The SPRHW CI POSIX.2 compliant platform shall have the following POSIX.2 Software Development Utilities installed at a minimum: make.	SPRHW CI;
S-DPS-61150	The SPRHW CI POSIX.2 compliant platform shall have the following POSIX.2 C-Language Development Utilities installedat a minimum: lex, yacc.	SPRHW CI;
S-DPS-61160	The SPRHW CI POSIX.2 compliant platform shall have the following Unix shells installed at a minimum: C shell, Bourne shell, Korn shell.	SPRHW CI;

Table A-1. Requirements Trace (25 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-61170	The SPRHW CI POSIX.2 compliant platform shall have online documentation or printed documentation for each installed tool.	SPRHW CI;
S-DPS-61171	The SPRHW CI shall have provision for a dynamic analyzer to support the capability to check Science Software source code for memory leaks.	SPRHW CI;
S-DPS-61172	The SPRHW CI POSIX.2 compliant platform shall have installed one or more development environment supporting the following languages: a. C b. C++ c. FORTRAN 77 d. FORTRAN 90	SPRHW CI;
S-DPS-61173	Each development environment associated with the POSIX.2 compliant platform in the SPRHW CI shall have the capability to compile and link strictly conformant POSIX-compliant source code.	SPRHW CI;
S-DPS-61174	Each development environment associated with the POSIX.2 compliant platform in the SPRHW CI shall have the capability to compile and link source code containing extensions specified in the Data Production S/W and SCF Standards and Guidelines.	SPRHW CI;
S-DPS-61175	Each development environment associated with the POSIX.2 compliant platform in the SPRHW CI shall have an interactive source level debugger for ECS supported languages.	SPRHW CI;
S-DPS-61177	The SPRHW CI POSIX.2 compliant platform supporting AI&T of CERES S/W shall have installed an ADA development environment.	SPRHW CI;
S-DPS-70010	The AITHW CI shall provide hardware resources to operations staff for the monitor and control of Science Software Integration and Test (AI&T) on SPRHW CI processing resources.	AITHW CI;
S-DPS-70030	The AITHW CI shall provide hardware resources to operations staff for the monitor and control of Science Software configuration management.	AITHW CI;
S-DPS-70030	The AITHW CI shall provide hardware resources to operations staff for the monitor and control of Science Software configuration management.	AITHW CI;
S-DPS-70050	The Algorithm Integration and Test HWCI design and implementation shall have the flexibility to accommodate Algorithm Integration and Test expansion up to a factor of 3 in its capacity with no changes in its design and up to a factor of 10 without major changes to its design.	AITHW CI;
S-DPS-70060	The AITHW CI shall have provision for Initialization, Recovery, and an orderly shutdown.	AITHW CI;
S-DPS-70070	The AITHW CI shall have a status monitoring capability.	AITHW CI;

Table A-1. Requirements Trace (26 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-70110	The operating system for each UNIX platform in the AITHW CI shall conform to the POSIX.2 standard.	AITHW CI;
S-DPS-70120	The AITHW CI POSIX.2 compliant platform shall have the following utilities installed at a minimum: perl, emacs, gzip, tar, imake, prof, gprof, nm.	AITHW CI;
S-DPS-70130	The AITHW CI POSIX.2 compliant platform ishall have the following POSIX.2 User Portability Utilities installed at a minimum: man, vi.	AITHW CI;
S-DPS-70140	The AITHW CI POSIX.2 compliant platform shall have the following POSIX.2 Software Development Utilities installed at a minimum: make.	AITHW CI;
S-DPS-70150	The AITHW CI POSIX.2 compliant platform shall have the following POSIX.2 C-Language Development Utilities installed at a minimum: lex, yacc.	AITHW CI;
S-DPS-70160	The AITHW CI POSIX.2 compliant platform shall have the following Unix shells installed at a minimum: C shell, Bourne shell, Korn shell.	AITHW CI;
S-DPS-70180	The AITHW CI shall have provision for a dynamic analyzer to support the capability to check Science Software source code for memory leaks.	AITHW CI;
S-DPS-70183	The AITHW CI POSIX.2 compliant platform shall have online documentation or printed documentation for each installed tool.	AITHW CI;
S-DPS-70190	The AITHW CI POSIX.2 compliant platform shall have installed one or more development environment supporting the following languages: a. C b. C++ c. FORTRAN 77 d. FORTRAN 90	AITHW CI;
S-DPS-70220	Each development environment associated with the POSIX.2 compliant platform in the AITHW CI shall have the capability to compile and link strictly conformant POSIX-compliant source code.	AITHW CI;
S-DPS-70230	Each development environment associated with the POSIX.2 compliant platform in the AITHW CI shall have the capability to compile and link source code containing extensions specified in the Data Production S/W and SCF Standards and Guidelines.	AITHW CI;
S-DPS-70240	Each development environment associated with the POSIX.2 compliant platform in the AITHW CI shall have an interactive source level debugger for ECS supported languages.	AITHW CI;
S-DPS-70250	Each development environment associated with the POSIX.2 compliant platform in the AITHW CI shall have a screen capture utility.	AITHW CI;

Table A-1. Requirements Trace (27 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-70260	The AITHW CI shall include a set of profiling tools, with the capability to measure the average and maximum of the following: a. CPU time, b. memory usage, c. disk space usage of a process.	AITHW CI;
S-DPS-70270	The AITHW CI profiling tools shall be accessible via an API (application program interface).	AITHW CI;
S-DPS-70280	The AITHW CI profiling tools shall be accessible via a GUI (graphical user interface).	AITHW CI;
S-DPS-70310	The AITHW CI platforms shall have provision for interfacing with one or more Local Area Networks (LANs).	AITHW CI;
S-DPS-70710	The electrical power requirements for AITHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	AITHW CI;
S-DPS-70740	The air conditioning requirements for the AITHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	AITHW CI;
S-DPS-70750	The grounding requirements for AITHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	AITHW CI;
S-DPS-70760	The fire alarm requirements for AITHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	AITHW CI;
S-DPS-70770	The acoustical requirements for AITHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	AITHW CI;
S-DPS-70780	The physical interface requirements between AITHW CI equipment and the facility shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	AITHW CI;
S-DPS-70790	The footprint size and the physical layout of AITHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	AITHW CI;
S-DPS-80010	The AQAHW CI shall provide for hardware resources to support DAAC operations staff performing routine QA of Product data.	AQAHW CI;
S-DPS-80011	The AQAHW CI shall provide an operational availability of TBD , at a minimum, and an MDT of TBDhours or less.	AQAHW CI;
S-DPS-80110	The operating system for each UNIX platform in the AQAHW CI shall conform to the POSIX.2 standard.	AQAHW CI;
S-DPS-80120	The AQAHW CI POSIX.2 compliant platform shall have the following utilities installed at a minimum: perl, emacs, gzip, tar, imake, prof, gprof, nm.	AQAHW CI;
S-DPS-80130	The AQAHW CI POSIX.2 compliant platform ishall have the following POSIX.2 User Portability Utilities installed at a minimum: man, vi.	AQAHW CI;
S-DPS-80140	The AQAHW CI POSIX.2 compliant platform shall have the following POSIX.2 Software Development Utilities installed at a minimum: make.	AQAHW CI;

Table A-1. Requirements Trace (28 of 28)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-DPS-80150	The AQAHW CI POSIX.2 compliant platform shall have the following Unix shells installed at a minimum: C shell, Bourne shell, Korn shell.	AQAHW CI;
S-DPS-80155	The AQAHW CI POSIX.2 compliant platform shall have on-line documentation or printed documentation for each installed tool.	AQAHW CI;

NOTES:

1. The requirements for the DPREP CSCI are being revised to reflect their allocation to PRONG and INS CSCIs. The DPREP requirements stated here reflects those requirements prior to the rewording and includes only those requirments allocated to PRONG CSCI for Release A.

Abbreviations and Acronyms

ADSRV	advertising service CSCI
AHWGP	Ad Hoc Working Group on Production
AI&T	algorithm integration and test
AITTL	algorithm integration and test tools (CSCI)
AM-1	EOS AM Project (morning spacecraft series)
ASCII	American Standard Code for Information Exchange Interchange
CASE	computer aided software engineering
CDR	Critical Design Review
CERES	Clouds and Earth's Radiant Energy System
CI	configuration item
COTS	commercial off-the-shelf (hardware or software)
CPU	central processing unit
CSC	computer software component
CSCI	computer science configuration item
CSMS	Communications and Systems Management Segment (ECS)
DAAC	distributed active archive center
DBMS	database management system
DDSRV	document data server
DEV	developed code
DPR	Data Processing Request
DPS	Data Processing Subsystem
ECS	EOSDIS Core System
EDF	ECS development facility
ESDT	Earth science data types
ESN	EOSDIS Science Network (ECS)
FDDI	fiber distributed data interface
GB	gigabyte (10^9)
GC	global change
GUI	graphic user interface
HCL	Hughes class library
HTML	Hyper-Text Markup Language

HWCI	hardware configuration item
I/O	input/output
L0	Level 0
LAN	local area network
LaRC	Langley Research Center (DAAC)
MB	megabyte (10^6)
MSFC	Marshall Space Flight Center (DAAC)
MSS	Management Subsystem
NOAA	National Oceanic and Atmospheric Administration
OO	object oriented
PDPS	planning and data processing system
PDR	Preliminary Design Review
PGE	product generation executive (formerly product generation executable)
PLANG	production planning CSCI
PRONG	processing CSCI
RAID	redundant array of inexpensive disks
RDBMS	relational database management system
RMA	reliability, maintainability, availability
SCSI	Small Computer System Interface
SDPF	Sensor Data Processing Facility (GSFC)
SDPS	Science Data Processing Segment (ECS)
TRMM	Tropical Rainfall Measuring Mission (joint US-Japan)
U/I	user interface
UNIX	POSIX operating system
WAIS	Wide Area Information Server
WWW	World Wide Web